버전 2.x용 개발자 가이드

AWS SDK for Java 2.x

AWS SDK for Java 2.x: 버전 2.x용 개발자 가이드

Table of Contents

란 무엇입니까? AWS SDK for Java 2.x

는 용 Java API를 AWS SDK for Java 제공합니다 AWS 서비스. SDK를 사용하면 Amazon S3 Amazon EC2 DynamoDB, 등으로 작동하는 Java 애플리케이션을 빌드할 수 있습니다.

AWS SDK for Java 2.x 는 버전 1.x 코드 베이스의 주요 재작성입니다. Java 8+에 토대를 두고 있으며, 요청이 많았던 기능들을 몇 가지 추가했습니다. 여기에는 비차단 I/O에 대한 지원과 런타임에 다른 HTTP 구현을 연결하는 기능이 포함됩니다.

Amazon에서는 새 서비스에 대한 지원을 AWS SDK for Java에 정기적으로 추가하고 있습니다. 특정 버전의 변경 사항 및 기능 목록은 변경 로그를 참조하세요.

SDK 시작하기

SDK를 직접 사용해 볼 준비가 되었다면 이 시작하기 자습서를 참고하세요.

개발 환경을 설정하려면 the section called “설정”을 참조하세요.

현재 버전 1.x를 사용하는 경우 SDK for Java특정 지침은 버전 2로 마이그레이션을 참조하세요.

요청 및 Amazon S3 DynamoDB Amazon EC2 기타 요청에 대한 자세한 내용은 사용 SDK for Java 및 작업을 AWS 서비스 AWS 서비스참조하세요.

모바일 애플리케이션 개발

모바일 앱 개발자인 경우 AWS Amplify 프레임워크를 Amazon Web Services 제공합니다.

SDK 메이저 버전에 대한 유지 관리 및 지원

SDK 메이저 버전 및 기본 종속성의 유지 관리 및 지원에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서에서 다음 내용을 참조하세요.

- AWS SDKs 및 도구 유지 관리 정책
- AWS SDKs 및 도구 버전 지원 매트릭스

추가 리소스

이 가이드 외에도 개발 AWS SDK for Java 자에게 유용한 온라인 리소스는 다음과 같습니다.

- AWS SDK for Java 2.x API 참조
- Java 개발자 블로그
- 의 Java 개발 주제 AWS re:Post
- GitHub의 SDK 소스
- AWS SDK 코드 예제 라이브러리
- @awsforjava(Twitter)

SDK에 기여

개발자는 다음 채널을 통해 피드백을 제공할 수 있습니다.

- GitHub에서 SDK 문제 제출
- AWS SDK for Java 2.x gitter 채널에서 SDK에 대한 비공식 채팅에 참여합니다.

AWS SDK for Java 2.x로 시작하기

이 주제의 섹션에서는 AWS 서비스에 연결하는 Java 애플리케이션 구축을 시작하는 데 필요한 단계를 안내합니다. 이 섹션에서는 Java를 사용하여 개발 환경을 설정하고 Maven 또는 Gradle 등의 도구를 구축하고, AWS에 대한 보안 인증을 구성하고, 실습 자습서를 통해 첫 번째 작업 애플리케이션을 만드는 방법을 다룹니다. 이 초보자용 주제는 Java를 사용한 AWS 개발의 시작점 역할을 하여 고급 기능을 살펴보기 전에 필요한 기본 사항을 제공합니다.

목차

AWS SDK for Java 2.x 설정

이 단원에서는 AWS SDK for Java 2.x을 사용하기 위한 개발 환경을 설정하는 방법에 대한 정보를 제공합니다.

설정 개요

AWS SDK for Java를 사용하여AWS 서비스에 액세스하는 애플리케이션을 성공적으로 개발하려면 다음 조건이 필요합니다.

- 사용자를 대신하여 요청을 인증하려면 Java SDK가 자격 증명에 액세스할 수 있어야 합니다.
- SDK에 구성된 IAM 역할의 권한은 애플리케이션에 필요한 AWS 서비스 액세스를 허용해야 합니다. PowerUserAccess AWS 관리형 정책과 관련된 권한은 대부분의 개발 요구 사항에 충분합니다.
- 다음 요소가 포함된 개발 환경
  - 최소 다음 중 하나와 같은 방식으로 설정된 공유 구성 파일:
    - 이 `config` 파일에는 SDK가 AWS 자격 증명을 가져올 수 있도록 IAM Identity Center 싱글 사인온 설정이 포함되어 있습니다.
    - 이 `credentials` 파일에는 임시 자격 증명이 들어 있습니다.
  - Java 8 이상 버전 설치.
  - Maven 또는 Gradle과 같은 빌드 자동화 도구.
  - 코드 작업을 위한 텍스트 편집기.
  - (선택 사항이지만 권장됨) IntelliJ IDEA, Eclipse 또는 NetBeans와 같은 IDE(통합 개발 환경).

    또한 IntelliJ IDEA를 사용하는 경우 AWS Toolkit for IntelliJ IDEA를 추가하여 IDE에 AWS 서비스 직접 통합하여 개발을 간소화할 수 있습니다.

- 애플리케이션을 실행할 준비가 되었을 때 활성화된 AWS 액세스 포털 세션. AWS Command Line Interface를 사용하여 IAM Identity Center의 AWS 액세스 포털에 대한 로그인 프로세스를 시작합니다.

> **Important**
>
> 이 설정 단원의 지침에서는 사용자 또는 조직이 IAM Identity Center를 사용한다고 가정합니다. 조직에서 IAM Identity Center와 독립적으로 작동하는 외부 ID 공급자를 사용하는 경우 Java용 SDK에 사용할 임시 자격 증명을 얻는 방법을 알아보세요. 이 지침에 따라 `~/.aws/credentials` 파일에 임시 자격 증명을 추가하세요.
> ID 제공자가 임시 자격 증명을 `~/.aws/credentials` 파일에 자동으로 추가하는 경우 SDK 또는 AWS CLI 에 프로필 이름을 제공할 필요가 없도록 프로필 이름을 `[default]`으로 지정해야 합니다.

AWS SDK for Java 2.x에서 사용할 Java 및 빌드 도구 설치

SDK for Java 2.x로 작업하려면 다음과 같은 Java 개발 환경 요구 사항이 필요합니다.

- Java 8 이상. AWS SDK for Java는 Oracle Java SE 개발 키트와 Amazon Corretto, Red Hat OpenJDK, Adoptium 등의 오픈 Java 개발 키트(OpenJDK) 배포판과 함께 작동합니다.
- Apache Maven, Ivy, Gradle 또는 IntelliJ를 사용하는 Apache Ant 등 Maven 호환 빌드 도구입니다.
  - Maven 설치 및 사용 방법에 대한 자세한 내용은 https://maven.apache.org/을 참조하세요.
  - Apache Ivy 설치 및 사용 방법에 대한 자세한 내용은 https://ant.apache.org/ivy/ 페이지를 참조하세요.
  - Gradle 설치 및 사용 방법에 대한 자세한 내용은 https://gradle.org/을 참조하세요.
  - IntelliJ IDEA 설치 및 사용 방법에 대한 자세한 내용은 https://www.jetbrains.com/idea/을 참조하세요.

AWS SDK for Java 2.x를 사용하는 Apache Maven 프로젝트 설정

Apache Maven을 사용하여 AWS SDK for Java 2.x 프로젝트를 구성 및 빌드하거나 SDK 자체를 빌드할 수 있습니다.

사전 조건

Maven에서 SDK for Java 2.x를 사용하려면 다음이 필요합니다.

- Java 8.0 이상. http://www.oracle.com/technetwork/java/javase/downloads/에서 최신 Java SE Development Kit 소프트웨어를 다운로드할 수 있습니다. 또한 SDK for Java 2.x는 OpenJDK 및 오픈 Java 개발 키트(OpenJDK) 배포인 Amazon Corretto와 작동합니다. https://openjdk.java.net/install/index.html에서 최신 OpenJDK 버전을 다운로드하십시오. Corretto 페이지에서 최신 Amazon Corretto 8 또는 Amazon Corretto 11 버전을 다운로드하세요.
- – Apache Maven Maven을 설치해야 하는 경우 http://maven.apache.org/에서 다운로드하여 설치하십시오.

Maven 프로젝트 만들기

명령줄에서 Maven 프로젝트를 만들려면 터미널 또는 명령 프롬프트 창에서 다음 명령을 실행합니다.

```
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**
>
> com.example.myapp을 애플리케이션의 전체 패키지 네임스페이스로 바꿉니다. 또한 myapp을 프로젝트 이름으로 바꿉니다. 이는 프로젝트의 디렉터리 이름이 됩니다.
> 최신 버전의 아키타입을 사용하려면 *2.X.X*를 Maven Central의 최신 버전으로 바꾸세요.

이 명령은 아키타입 템플릿 툴킷을 사용하여 Maven 프로젝트를 만듭니다. 아키타입은 AWS Lambda 함수 핸들러 프로젝트를 위한 스캐폴딩을 생성합니다. 이 프로젝트 아키타입은 Java SE 8로 컴파일하도록 미리 구성되어 있으며 `-DarchetypeVersion`로 지정된 Java 2.x용 SDK 버전에 대한 종속성을 포함합니다.

Maven 프로젝트 생성 및 구성에 대한 자세한 내용은 Maven 시작 안내서를 참조하십시오.

Maven에 Java 컴파일러 구성

앞에서 설명한 대로 AWS Lambda 프로젝트 아키타입을 사용하여 프로젝트를 만든 경우에는 Java 컴파일러 구성이 이미 완료되어 있습니다.

이러한 구성이 있는지 확인하려면, 이전 명령을 실행했을 때 만든 프로젝트 폴더(예: `pom.xml`)에서 `myapp` 파일을 엽니다. 이 Maven 프로젝트의 Java 컴파일러 버전 설정과 71~75행에 Maven 컴파일러 플러그인을 포함시키는 데 필요한 내용을 보려면 11행 및 12행을 살펴보십시오.

```
<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>
```

다른 아키타입이나 다른 방법을 사용하여 프로젝트를 생성하는 경우 Maven 컴파일러 플러그인이 빌드의 일부이고 해당 소스 및 대상 속성이 모두 `pom.xml` 파일에서 1.8로 설정되어 있는지 확인해야 합니다.

이러한 필수 설정을 구성하는 한 가지 방법에 대해서는 이전 조각을 참조하십시오.

또는 다음과 같이 플러그인 선언을 통해 컴파일러 구성 인라인을 구성할 수 있습니다.

```
<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>
```

SDK를 종속성으로 선언

AWS SDK for Java를 프로젝트에서 사용하려면 `pom.xml` 파일에서 종속성으로 선언해야 합니다.

앞에서 설명한 대로 프로젝트 아키타입를 사용하여 프로젝트를 만들었다면 SDK는 이미 프로젝트에서 종속성으로 구성되어 있습니다.

아키타입은 `software.amazon.awssdk` 그룹 ID에 대한 BOM(재료 명세서) 아티팩트 종속성을 생성합니다. BOM을 사용하면 동일한 그룹 ID를 공유하는 개별 아티팩트 종속성에 대해 Maven 버전을 지정할 필요가 없습니다.

다른 방법으로 Maven 프로젝트를 만든 경우에는 `pom.xml` 파일에 다음 사항이 포함되어 있는지 확인하여 프로젝트에 SDK의 최신 버전을 구성하십시오.

```
<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**
>
> `pom.xml` 파일의 *`2.X.X`*를 AWS SDK for Java 2.x의 최신 버전으로 교체하세요.

SDK 모듈에 대한 종속성 설정

SDK를 구성했으므로 프로젝트에서 사용할 하나 이상의 AWS SDK for Java 모듈에 종속성을 추가할 수 있습니다.

각 구성 요소의 버전 번호를 지정할 수 있지만 이미 BOM 아티팩트를 사용하여 `dependencyManagement` 섹션에서 SDK 버전을 선언했기 때문에 그럴 필요가 없습니다. 지정된 모듈의 다른 버전을 로드하려면 해당 종속성에 버전 번호를 지정합니다.

앞에서 설명한 대로 프로젝트 아키타입를 사용하여 프로젝트를 만들었다면 프로젝트는 이미 여러 종속성으로 구성되어 있습니다. 여기에는 다음과 같이 AWS Lambda 함수 핸들러와 Amazon S3에 대한 종속성이 포함됩니다.

```
<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**
>
> 위 `pom.xml` 예시에서는 종속성이 서로 다른 `groupId`에 속합니다. `s3` 종속성은 `software.amazon.awssdk`에서 가져온 것이고 `aws-lambda-java-core` 종속성은 `com.amazonaws`에서 가져온 것입니다. BOM 종속성 관리 구성은 `software.amazon.awssdk`의 아티팩트에 영향을 미치므로 `aws-lambda-java-core` 아티팩트에 대한 버전이 필요합니다.
> Java 2.x용 SDK를 사용하여 Lambda 함수 핸들러를 개발하는 경우 `aws-lambda-java-core`가 올바른 종속성입니다. 하지만 `listFunctions`, `deleteFunction`, `invokeFunction`, `createFunction` 등의 작업을 사용하여 애플리케이션이 Lambda 리소스를 관리해야 하는 경우 애플리케이션에 다음과 같은 종속성이 필요합니다.
>
> ```
> <groupId>software.amazon.awssdk</groupId>
> <artifactId>lambda</artifactId>
> ```

> **Note**
>
> `s3` 종속성에는 `netty-nio-client` 및 `apache-client` 전이적 종속성이 제외됩니다. 아키타입에는 이러한 HTTP 클라이언트 대신 `url-connection-client` 종속성이 포함되므로 AWS Lambda 함수의 시작 지연 시간을 줄이는 데 도움이 됩니다.

프로젝트에 필요한 AWS 서비스 및 기능을 위해 프로젝트에 모듈을 추가합니다. AWS SDK for Java BOM에서 관리하는 모듈(종속성)은 Maven 중앙 저장소에 나열되어 있습니다.

> **Note**
>
> 코드 예제에서 `pom.xml` 파일을 살펴보고 프로젝트에 필요한 종속성을 결정할 수 있습니다. 예를 들어 DynamoDB 서비스의 종속성에 관심이 있다면 GitHub의 AWS 코드 예제 리포지토리에서 이 예제를 참조하세요. (/javav2/example_code/dynamodb에서 `pom.xml` 파일을 찾아보세요.)

전체 SDK를 프로젝트에 빌드

애플리케이션을 최적화하려면 전체 SDK 대신 필요한 구성 요소만 가져오는 것이 좋습니다. 그러나 전체 AWS SDK for Java를 프로젝트에 빌드하려면 다음과 같이 `pom.xml` 파일에 선언해야 합니다.

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

프로젝트 빌드

pom.xml 파일을 구성한 후에는 Maven을 사용하여 프로젝트를 빌드할 수 있습니다.

명령줄에서 Maven 프로젝트를 빌드하려면 터미널 또는 명령 프롬프트 창을 열고 프로젝트 디렉터리(예: myapp)로 이동하여 다음 명령을 입력하거나 붙여넣은 다음 Enter 또는 Return을 누릅니다.

```
mvn package
```

이렇게 하면 .jar 디렉터리(예: target)에 단일 myapp/target 파일(JAR)이 생성됩니다. 이 JAR에는 pom.xml 파일에서 종속성으로 지정한 모든 SDK 모듈이 들어 있습니다.

AWS SDK for Java 2.x를 사용하는 Gradle 프로젝트 설정

Gradle을 사용하여 AWS SDK for Java 2.x 프로젝트를 설정하고 빌드할 수 있습니다.

다음 예시의 초기 단계는 Gradle 버전 8.4용 시작 가이드에서 가져온 것입니다. 다른 버전을 사용하는 경우 결과가 약간 다를 수 있습니다.

Gradle을 사용하여 Java 애플리케이션을 만들려면(명령줄)

1. 프로젝트를 보관할 디렉터리를 생성합니다. 이 예에서 디렉터리 이름은 demo입니다.
2. demo 디렉터리 내에서 gradle init 명령을 실행하고 다음 명령줄 출력과 같이 빨간색으로 강조 표시된 값을 입력합니다. 안내에서는 Kotlin을 빌드 스크립트 DSL 언어로 선택했지만 Groovy에 대한 전체 예제도 이 항목의 끝에 표시됩니다.

```
> 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. `init` 작업이 완료되면 demo 디렉터리에는 다음과 같은 트리 구조가 포함됩니다. 다음 단원에서는 주 빌드 파일 `build.gradle.kts`(빨간색으로 강조 표시됨)을 자세히 살펴보겠습니다.

```
### 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
```

build.gradle.kts 파일에는 다음의 스캐폴드 콘텐츠가 포함되어 있습니다.

```
/*
 * 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:33.3.0-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. 스캐폴딩된 Gradle 빌드 파일을 AWS 프로젝트의 기반으로 사용하세요.

   a. Gradle 프로젝트에 대한 SDK 종속성을 관리하려면 AWS SDK for Java 2.x용 Maven BOM(Bill of Materials)을 `build.gradle.kts` 파일의 `dependencies` 섹션으로 가져옵니다.

```
...
dependencies {
    implementation(platform("software.amazon.awssdk:bom:2.27.21"))
    // With the bom declared, you specify individual SDK dependencies without a
 version.
 ...
}
...
```

> **Note**
>
> 이 예제 빌드 파일에서 2.27.21을 Java 2.x용 SDK의 최신 버전으로 바꾸세요. Maven 중앙 리포지토리에서 최신 버전을 찾아보세요.

b. `dependencies` 섹션에서 애플리케이션에 필요한 SDK 모듈을 지정하세요. 예를 들어, 다음은 Amazon Simple Storage Service에 대한 종속성을 추가합니다.

```
...
dependencies {
    implementation(platform("software.amazon.awssdk:bom:2.27.21"))
    implementation("software.amazon.awssdk:s3")
 ...
}
...
```

Gradle은 BOM의 정보를 사용하여 선언된 종속성의 올바른 버전을 자동으로 확인합니다.

다음 예시는 Kotlin과 Groovy DSL의 전체 Gradle 빌드 파일을 보여줍니다. 빌드 파일에는 Amazon S3에 대한 종속성, 인증, 로깅 및 테스트가 포함되어 있습니다. Java의 소스 및 대상 버전은 버전 11입니다.

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:2.27.21')
    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()
```

```
}
```

다음 단계는 Gradle 웹사이트의 시작 가이드를 참고하여 Gradle 애플리케이션을 빌드하고 실행하는 방법에 대한 지침을 확인하세요.

AWS SDK for Java 2.x를 사용한 GraalVM Native Image 프로젝트 설정

버전 2.16.1 이상에서는 AWS SDK for Java 2.x에서 GraalVM Native Image 애플리케이션을 기본적으로 지원합니다. `archetype-app-quickstart` Maven 아키타입을 사용하여 네이티브 이미지 지원이 내장된 프로젝트를 설정할 수 있습니다.

사전 조건

- AWS SDK for Java 2.x 설정의 단계를 완료하세요.
- GraalVM Native Image를 설치합니다.

아키타입을 사용하여 프로젝트 생성

기본 제공 네이티브 이미지 지원을 사용하여 Maven 프로젝트를 생성하려면 터미널 또는 명령 프롬프트 창에서 다음 명령을 사용합니다.

> **Note**
>
> `com.example.mynativeimageapp`를 애플리케이션의 전체 패키지 네임스페이스로 바꿉니다. 또한 `mynativeimageapp`을 프로젝트 이름으로 바꿉니다. 이는 프로젝트의 디렉터리 이름이 됩니다.

```
mvn archetype:generate \
    -DarchetypeGroupId=software.amazon.awssdk \
    -DarchetypeArtifactId=archetype-app-quickstart \
    -DarchetypeVersion=2.27.21\
    -DnativeImage=true \
    -DhttpClient=apache-client \
    -Dservice=s3 \
    -DgroupId=com.example.mynativeimageapp \
    -DartifactId=mynativeimageapp \
    -DinteractiveMode=false
```

이 명령은 AWS SDK for Java, Amazon S3, 및 ApacheHttpClient HTTP 클라이언트에 대한 종속성을 사용하여 구성된 Maven 프로젝트를 만듭니다. 또한 GraalVM Native Image Maven 플러그인에 대한 종속성도 포함되어 있으므로 Maven을 사용하여 네이티브 이미지를 빌드할 수 있습니다.

다른 Amazon Web Services의 종속성을 포함하려면 -Dservice 파라미터 값을 해당 서비스의 아티팩트 ID로 설정하세요. 예를 들면 dynamodb, comprehend 및 pinpoint입니다. 아티팩트 ID의 전체 목록은 Maven Central의 software.amazon.awssdk에 대한 관리형 종속성 목록을 참조하세요.

비동기 HTTP 클라이언트를 사용하려면 -DhttpClient 파라미터를 netty-nio-client로 설정하세요. apache-client 대신 UrlConnectionHttpClient를 동기 HTTP 클라이언트로 사용하려면 -DhttpClient 파라미터를 url-connection-client로 설정합니다.

네이티브 이미지를 빌드

프로젝트를 생성한 후 프로젝트 디렉터리에서 예로 mynativeimageapp와 같은 다음 명령어를 실행합니다.

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

그러면 target 디렉터리에 예로 target/mynativeimageapp와 같은 네이티브 이미지 애플리케이션이 생성됩니다.

를 AWS 사용하여 로 인증 AWS SDK for Java 2.x

AWS 서비스를 사용하여 개발할 AWS 때가를 AWS SDK for Java 2.x 인증하는 방법을 설정해야 합니다. SDK는 자격 증명 검색, 서명 작성 및 자격 증명 새로 고침을 백그라운드에서 완전히 관리하므로 애플리케이션 로직에 중점을 둘 수 있습니다.

인증 설정

AWS SDKs 및 도구 참조 안내서의 인증 및 액세스 주제에서는 다양한 인증 접근 방식을 설명합니다.

로컬 개발의 경우 콘솔 로그인 자격 증명으로 AWS CLI에 로그인하여 단기 자격 증명을 사용하는 것이 좋습니다. AWS 계정 액세스를 위해 루트, IAM 사용자 또는 IAM과의 페더레이션을 사용하는 경우 권장됩니다. AWS SDK for Java 2.x를 사용한 대화형 개발 작업용 액세스 자격 증명의 지침을 따릅니다.

AWS SDKs 및 도구 참조 안내서의 지침에 따라 SDK가 요청에 서명할 수 있도록 시스템을 설정해야 합니다.

1. 단기 자격 증명을 사용한 로컬 개발

로컬 개발의 경우 콘솔 로그인 자격 증명으로 AWS CLI에 로그인하여 단기 자격 증명을 사용하는 것이 좋습니다. AWS 계정 액세스를 위해 루트, IAM 사용자 또는 IAM과의 페더레이션을 사용하는 경우 권장됩니다.

지침은 의 자격 증명 공급자 사용을 참조하세요 AWS SDK for Java 2.x.

2. SDK에 Single Sign-On 액세스 설정

SDK가 IAM Identity Center 인증을 사용하도록 하려면 프로그래밍 방식 액세스 섹션의 2단계를 완료한 후, 시스템에 다음 요소가 포함되도록 해야 합니다.

- 애플리케이션을 실행하기 전에 AWS 액세스 포털 세션을 시작하는 데 AWS CLI사용하는 입니다.
- 기본 프로필이 포함된 `~/.aws/config` 파일. Java용 SDK는 AWS에 요청을 보내기 전에 프로필의 SSO 토큰 공급자 구성를 사용하여 보안 인증을 얻습니다. `sso_role_name` 값은 IAM 신원 센터 권한 집합에 연결된 IAM 역할로, 애플리케이션에서 사용되는 AWS 서비스 서비스에 대한 액세스를 허용해야 합니다.

  다음 샘플 `config` 파일은 SSO 토큰 공급자 구성으로 설정된 기본 프로필을 보여줍니다. 프로필의 `sso_session` 설정은 이름이 지정된 `sso-session` 섹션을 참조합니다. `sso-session` 섹션에는 AWS 액세스 포털 세션을 시작하기 위한 설정이 포함되어 있습니다.

```
[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
```

SSO 토큰 공급자 구성에 사용되는 설정에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서의 SSO 토큰 공급자 구성을 참조하세요.

개발 환경이 전에 표시된 것처럼 프로그래밍 방식으로 액세스할 수 있도록 설정되지 않은 경우 SDK 참조 가이드의 2단계를 따르세요.

3. 를 사용하여 로그인 AWS CLI

에 액세스하는 애플리케이션을 실행하기 전에 SDK가 IAM Identity Center 인증을 사용하여 자격 증명을 확인하려면 활성 AWS 액세스 포털 세션이 AWS 서비스필요합니다. 에서 다음 명령을 실행 AWS CLI 하여 AWS 액세스 포털에 로그인합니다.

```
aws sso login
```

기본 프로필이 설정되어 있으므로 `--profile` 옵션으로 명령을 호출할 필요가 없습니다. SSO 토큰 공급자 구성에서 명명된 프로필을 사용하는 경우 `aws sso login --profile named-profile` 명령을 사용합니다.

이미 활성 세션이 있는지 테스트하려면 다음 AWS CLI 명령을 실행합니다.

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

이 명령에 대한 응답은 공유 `config` 파일에 구성된 IAM Identity Center 계정 및 권한 집합을 보고해야 합니다.

> **Note**
>
> 이미 활성 AWS 액세스 포털 세션이 있고를 실행하는 경우 `aws sso login`자격 증명을 제공할 필요가 없습니다.
> 하지만 정보에 액세스할 수 있는 `botocore` 권한을 요청하는 대화 상자가 표시됩니다.
> `botocore`는 AWS CLI 기반이 됩니다.
> 허용을 선택하여 AWS CLI 및 SDK for Java에 대한 정보에 대한 액세스를 승인합니다.

추가 인증 옵션

프로필 및 환경 변수 사용과 같은 SDK 인증에 대한 추가 옵션은 SDK 및 도구 참조 안내서의 구성 AWS SDKs.

AWS SDK for Java 2.x를 사용하여 간단한 애플리케이션 만들기

이 자습서에서는 Apache Maven을 사용하여 Java 2.x용 SDK의 종속성을 정의한 다음 파일을 업로드하기 위해 Amazon S3에 연결하는 코드를 작성하는 방법을 보여줍니다.

자습서를 완료하려면 이 단계를 따릅니다.

1단계: 튜토리얼 설정

이 튜토리얼을 시작하기 전에 다음이 필요합니다.

- Amazon S3 액세스 권한
- AWS IAM Identity Center에 대한 SSO(Single Sign-On)를 사용하여 AWS 서비스에 액세스할 수 있도록 구성된 Java 개발 환경

???의 지침을 사용하여 이 자습서를 설정하세요. Java SDK에 대한 Single Sign-On 액세스로 개발 환경을 구성하고 활성 AWS 액세스 포털 세션을 설정한 후 이 자습서의 2단계를 계속 진행하세요.

2단계: 프로젝트 생성

이 자습서의 프로젝트를 생성하려면 프로젝트 구성 방법에 대한 입력을 요청하는 Maven 명령을 실행합니다. 모든 입력이 입력되고 확인되면 Maven은 `pom.xml`를 생성하여 프로젝트 빌드를 완료하고 스텁 Java 파일을 생성합니다.

1. 터미널 또는 명령 프롬프트 창을 열고 원하는 디렉터리 (예: `Desktop` 또는 `Home` 폴더)로 이동합니다.
2. 터미널에서 다음 명령을 입력하고 `Enter` 키를 누릅니다.

```
mvn archetype:generate \
  -DarchetypeGroupId=software.amazon.awssdk \
  -DarchetypeArtifactId=archetype-app-quickstart \
  -DarchetypeVersion=2.27.21
```

3. 각 프롬프트의 두 번째 열에 나열된 값을 입력합니다.

프롬프트	입력할 값
`Define value for property 'service':`	`s3`

프롬프트	입력할 값
`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. 마지막 값을 입력하면 Maven은 사용자가 선택한 항목을 나열합니다. *Y*을 입력하여 확인하거나 *N*을 입력하여 값을 다시 입력합니다.

Maven은 입력한 `artifactId` 값을 기반으로 이름이 `getstarted`로 지정된 프로젝트 폴더를 만듭니다. `getstarted` 폴더 안에서 검토할 수 있는 `README.md` 파일, `pom.xml` 파일, `src` 디렉터리를 찾으세요.

Maven은 다음과 같은 디렉터리 트리를 만듭니다.

```
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
```

다음은 pom.xml 프로젝트 파일의 콘텐츠를 보여줍니다.

pom.xml

dependencyManagement 단원은 AWS SDK for Java 2.x대한 종속성을 포함하며 dependencies 섹션에는 Amazon S3에 대한 종속성이 있습니다. 프로젝트는 maven.compiler.source 및 maven.compiler.target 속성의 1.8 값 때문에 Java 1.8을 사용합니다.

```
<?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>2.27.21</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>
```

3단계: 코드 작성

다음 코드는 Maven이 생성한 App 클래스를 보여줍니다. main 메서드는 Handler 클래스의 인스턴스를 만든 다음 해당 sendRequest 메서드를 호출하는 애플리케이션의 진입점입니다.

App 클래스

```
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");
    }
}
```

Maven에서 만든 DependencyFactory 클래스에는 S3Client 인스턴스를 빌드하고 반환하는 s3Client 팩토리 메서드가 포함되어 있습니다. S3Client 인스턴스는 Apache 기반 HTTP 클라이언트의 인스턴스를 사용합니다. 이는 Maven에서 사용할 HTTP 클라이언트를 묻는 메시지가 표시될 때 사용자가 apache-client를 지정했기 때문입니다.

DependencyFactory는 다음 코드에 나와 있습니다.

DependencyFactory 클래스

```
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();
    }
}
```

Handler 클래스에는 프로그램의 기본 로직이 들어 있습니다. App 클래스에서 Handler 인스턴스가 생성되면 DependencyFactory는 S3Client 서비스 클라이언트를 제공합니다. 코드는 S3Client 인스턴스를 사용하여 Amazon S3 서비스를 호출합니다.

Maven은 *TODO* 주석과 함께 다음과 같은 Handler 클래스를 생성합니다. 자습서의 다음 단계에서는 *TODO*를 코드로 대체합니다.

Maven에서 생성한 **Handler** 클래스

```
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.
```

```
    }
}
```

로직을 채우려면 Handler 클래스의 전체 내용을 다음 코드로 바꾸세요. sendRequest 메서드가 채워지고 필요한 임포트가 추가됩니다.

Handler 클래스 구현됨

코드는 먼저 버킷 이름을 고유하게 만들기 위해 System.currentTimeMillis()를 사용하여 생성된 이름의 마지막 부분을 사용하여 새 S3 버킷을 만듭니다.

createBucket() 메서드에서 버킷을 생성한 후 프로그램은 S3Client의 putObject 메서드를 사용하여 객체를 업로드합니다. 객체의 내용은 RequestBody.fromString 메서드로 만든 간단한 문자열입니다.

마지막으로 프로그램은 cleanUp 메서드에서 객체를 삭제한 다음 버킷을 삭제합니다.

```
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");
    }
}
```

4단계: 애플리케이션 빌드 및 실행

프로젝트가 생성되고 전체 `Handler` 클래스가 포함된 후 애플리케이션을 빌드하고 실행합니다.

1. IAM IAM Identity Center 세션이 활성화되어 있는지 확인합니다. 이렇게 하려면 AWS Command Line Interface 명령 `aws sts get-caller-identity`을 실행하고 응답을 확인하세요. 활성 세션이 없는 경우 이 단원의 지침을 참조하세요.
2. 터미널 또는 명령 프롬프트 창을 열고 프로젝트 디렉토리 `getstarted`로 이동합니다.
3. 프로젝트를 빌드하려면 다음 명령을 사용합니다.

```
mvn clean package
```

4. 애플리케이션을 실행하려면 다음 명령을 사용합니다.

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

프로그램이 생성한 새 버킷과 객체를 보려면 다음 단계를 수행합니다.

1. `Handler.java`에서 `sendRequest` 메서드의 `cleanUp(s3Client, bucket, key)` 줄을 주석 처리한 다음 파일을 저장합니다.
2. `mvn clean package`를 실행하여 프로젝트를 다시 빌드합니다.
3. `mvn exec:java -Dexec.mainClass="org.example.App"`를 다시 실행하여 텍스트 객체를 한 번 더 업로드합니다.
4. S3 콘솔에 로그인하여 새로 생성된 버킷의 새 객체를 확인합니다.

파일을 확인한 후 객체를 삭제한 다음 버킷을 삭제합니다.

Success

Maven 프로젝트가 오류 없이 빌드되고 실행되었다면 축하합니다. Java 2.x용 SDK를 사용한 첫 Java 애플리케이션 구축에 성공했습니다.

정리

이 자습서를 진행하는 동안 생성한 리소스를 정리하려면 다음을 수행합니다.

- 아직 삭제하지 않았다면 S3 콘솔에서 애플리케이션을 실행할 때 생성된 모든 객체와 버킷을 삭제하세요.
- 프로젝트 폴더를 삭제합니다(`getstarted`).

다음 단계

이제 기본 사항을 갖추었으므로, 다음 내용을 배울 수 있습니다.

- 작업Amazon S3
- DynamoDB, Amazon EC2 및 다양한 데이터베이스 서비스등 다른 Amazon Web Services와 함께 작업하기
- SDK 사용하기
- AWS SDK for Java에서의 보안

AWS SDK for Java 2.x에서 서비스 클라이언트 구성

프로그래밍 방식으로 AWS 서비스에 액세스하기 위해 Java 2.x용 SDK는 각에 대해 클라이언트 객체를 사용합니다 AWS 서비스. 예를 들어 애플리케이션이 Amazon EC2에 액세스해야 하는 경우, Amazon EC2 클라이언트 객체(Ec2Client 클래스의 인스턴스)를 만들어 해당 서비스와 인터페이스합니다. 그런 다음 서비스 클라이언트를 사용하여 요청을 AWS 서비스에 보내면 됩니다. 대부분의 애플리케이션에서는 서비스 클라이언트의 싱글톤 인스턴스를 사용할 수 있습니다.

SDK 동작을 구성하는 방법에는 여러 가지가 있지만, 궁극적으로 모든 것은 서비스 클라이언트의 동작과 관련이 있습니다. 코드가 구성을 사용하는 서비스 클라이언트를 만들 때까지 모든 구성은 영향을 미치지 않습니다.

제공하는 구성의 예는 다음과 같습니다.

- 서비스를 호출할 AWS 때 로 코드를 인증하는 방법
- 서비스 클라이언트 AWS 리전 가 사용할
- 서비스 직접 호출 시 재시도 및 제한 시간 설정
- HTTP 프록시 구성

많은 SDK에 공통적인 설정, 기능 및 기타 기본 개념은 AWS SDKs 및 도구 참조 가이드를 참조하세요. AWS SDKs

주제

- AWS SDK for Java 2.x 외부에서 서비스 클라이언트 구성
- AWS SDK for Java 2.x에 대한 코드로 서비스 클라이언트 구성
- AWS SDK for Java 2.x에서 싱글톤 서비스 클라이언트 인스턴스 사용
- AWS SDK for Java 2.x에 대한 AWS 리전 설정
- 에서 자격 증명 공급자 사용 AWS SDK for Java 2.x
- AWS SDK for Java 2.x에서 재시도 동작 구성
- AWS SDK for Java 2.x에서 제한 시간 구성
- AWS SDK for Java 2.x에서 관찰성 기능 구성
- AWS SDK for Java 2.x에서 클라이언트 엔드포인트 구성

- 에서 HTTP 클라이언트 구성 AWS SDK for Java 2.x
- AWS SDK for Java 2.x에서 실행 인터셉터 사용

AWS SDK for Java 2.x 외부에서 서비스 클라이언트 구성

코드 외부에서 많은 구성 설정을 처리할 수 있습니다. 구성을 외부에서 처리할 때 동일한 Java 프로세스의 모든 애플리케이션에 적용할 수 있습니다. 대부분의 구성 설정은 환경 변수, JVM 시스템 속성 또는 별도의 공유 AWS `config` 파일로 설정할 수 있습니다. 공유 `config` 파일은 프로파일이라는 별도의 설정 세트를 유지하여 다양한 환경 또는 테스트에서 다양한 구성을 제공할 수 있습니다.

대부분의 환경 변수와 공유 `config` 파일 설정은 다양한 프로그래밍 언어 및 애플리케이션에서 일관된 기능을 지원하기 위해 AWS SDK 및 도구 간에 표준화되고 공유됩니다. 대부분의 경우 SDK for Java가 사용할 수 있는 JVM 시스템 속성은 환경 변수를 미러링합니다.

이러한 메서드 및 SDK 간 설정의 세부 사항을 통해 애플리케이션을 구성하는 방법을 알아보려면 AWS SDK 및 도구 참조 안내서를 참조하세요. 환경 변수, JVM 시스템 속성 또는 구성 파일에서 SDK가 해결할 수 있는 모든 설정을 보려면 AWS SDK 및 도구 참조 안내서에 나와 있는 설정 참조를 참조하세요.

클라이언트 구성용 구성 공급자 체인

SDK는 여러 위치(또는 소스)를 확인하여 구성 값을 찾습니다.

1. 코드나 서비스 클라이언트 자체에 설정된 모든 명시적 설정은 다른 모든 설정보다 우선합니다.
2. JVM 시스템 속성
   - JVM 시스템 속성 설정에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서에 나와 있는 JVM 시스템 속성을 설정하는 방법을 참조하세요.
3. 환경 변수
   - 환경 변수를 설정하는 방법에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서에 나와 있는 환경 변수 섹션을 참조하세요.
   - 시스템 전체, 사용자 전체, 특정 터미널 세션 등 다양한 수준의 쉘에 대한 환경 변수를 구성할 수 있습니다.
4. 공유 `config` 및 `credentials` 파일
   - 이 파일 설정에 관한 세부 사항을 알아보려면 AWS SDK 및 도구 참조 안내서에 나와 있는 공유 `config` 및 `credentials` 파일 섹션을 참조하세요.
5. SDK 소스 코드 자체에서 제공하는 모든 기본값이 마지막에 사용됩니다.

- 리전과 같은 일부 속성에는 기본값이 없습니다. 코드, 환경 설정 또는 공유 `config` 파일에서 명시적으로 지정해야 합니다. SDK가 필요한 구성을 해결할 수 없는 경우 API 요청은 런타임에 실패할 수 있습니다.

이 일반 구성 체인 외에도 SDK for Java 2.x는 자격 증명 공급자 체인 및 AWS 리전 공급자 체인을 포함한 특수 공급자 체인도 사용합니다. 이러한 특수 체인을 통해 SDK가 실행 중인 환경을 고려하는 추가 공급자를 추가합니다. 컨테이너 또는 EC2 인스턴스를 예로 들 수 있습니다.

외부 설정을 사용하여 구성된 서비스 클라이언트 만들기

AWS 서비스와 통신하려면 애플리케이션에서 서비스 클라이언트를 만들어야 합니다. 서비스 클라이언트는 AWS 서비스에 대한 필수 연결이며 복잡한 통신 세부 정보를 모두 처리하므로 걱정할 필요가 없습니다. 보안, 오류 처리 및 재시도와 같은 중요한 작업을 자동으로 처리하므로 기술적 복잡성을 처리하는 대신 애플리케이션 구축에 중점을 둘 수 있습니다.

`create()` 메서드 사용

필요한 모든 구성 설정을 외부 소스에서 가져온 경우 간단한 메서드로 서비스 클라이언트를 만들 수 있습니다.

```
S3Client s3Client = S3Client.create();
```

이전 코드 조각은 `S3Client` 인스턴스를 만듭니다. 만드는 중에 SDK는 구성 공급자 체인에서 설정을 찾습니다. SDK가 설정 값을 찾으면 나중에 체인에 있는 구성이 있더라도 값이 사용됩니다.

예를 들어 사용자가 시스템 속성 `-Daws.region=us-west-2`를 설정하여 AWS 리전에 대한 JVM 설정을 한다고 가정합니다. `AWS_REGION` 환경 변수도 설정되어 있으면 해당 값은 무시됩니다.

기본 리전 공급자 체인과 기본 자격 증명 공급자 체인도 만들기 프로세스에 사용됩니다. 체인의 어딘가에서 SDK는 사용할 AWS 리전을 해결하고 요청에 서명하기 위한 자격 증명을 검색할 수 있는 설정을 찾아야 합니다. SDK 파일에서 해당 값을 찾지 못하면 클라이언트를 만들 수 없습니다.

이 빈 빌더 패턴을 사용하여 클라이언트를 만들 수 있지만 코드에 구성을 추가하려는 경우 일반적으로 이 패턴을 사용합니다.

SDK for Java 2.x 환경 변수 및 JVM 시스템 속성

대부분의 AWS SDK에서 지원하는 cross-sdk 설정 외에도 SDK for Java 2.x는 다음 설정을 제공합니다.

> **Note**
>
> 이러한 환경 변수 및 JVM 시스템 속성은 주로 고급 사용 사례, 테스트 또는 특정 배포 시나리오를 위한 것입니다. 대부분의 애플리케이션 코드에서는 더 나은 유형 안전 및 IDE 지원을 위해 SDK의 클라이언트 빌더가 제공하는 프로그래밍 방식 구성 옵션을 사용하는 것이 좋습니다.

컨테이너 자격 증명 공급자 환경 변수

SDK는 참조 안내서에 설명된 표준 컨테이너 자격 증명 환경 변수 외에도 다음을 지원합니다.

`AWS_CONTAINER_SERVICE_ENDPOINT` - 이 환경 변수는 컨테이너 자격 증명 공급자를 사용할 때 컨테이너 메타데이터 서비스의 엔드포인트를 지정합니다.

Java 시스템 속성: `aws.containerServiceEndpoint`

기본 값: `http://169.254.170.2`

HTTP 클라이언트 구현 환경 변수

`SYNC_HTTP_SERVICE_IMPL` - SDK가 사용할 기본 동기식 HTTP 구현을 명시적으로 식별합니다. 클래스 경로에 여러 구현이 있거나 구현을 검색하는 데 클래스 경로 스캔이 필요하므로 성능 최적화가 필요한 경우에 유용합니다.

Java 시스템 속성: `software.amazon.awssdk.http.service.impl`

`ASYNC_HTTP_SERVICE_IMPL` - SDK가 사용할 기본 비동기식 HTTP 구현을 명시적으로 식별합니다. 클래스 경로에 여러 구현이 있거나 구현을 검색하는 데 클래스 경로 스캔이 필요하므로 성능 최적화가 필요한 경우에 유용합니다.

Java 시스템 속성: `software.amazon.awssdk.http.async.service.impl`

AWS SDK for Java 2.x에 대한 코드로 서비스 클라이언트 구성

서비스 클라이언트를 외부에서 구성하는 대신 코드에 프로그래밍 방식으로 구성할 수 있습니다.

코드에 서비스 클라이언트를 구성하여 사용 가능한 많은 옵션을 세밀하게 제어할 수 있습니다. 외부에서 설정할 수 있는 대부분의 구성도 코드에서 설정할 수 있습니다.

코드의 기본 구성

예를 들어 다음 코드 조각은 코드의 Amazon S3 서비스 클라이언트에 대해 AWS 리전을 EU_SOUTH_2로 설정합니다.

```
S3Client s3Client = S3Client.builder()
        .region(Region.EU_SOUTH_2)
        .build();
```

이전 코드 조각은 정적 팩토리 메서드(builder())를 보여줍니다. 이 builder() 메서드는 서비스 클라이언트를 사용자 지정할 수 있는 builder 객체를 반환합니다. 유용한 setter 메서드는 builder 객체(이 경우 S3ClientBuilder 인스턴스)를 반환하므로 편의성과 더 읽기 쉬운 코드를 위해 메서드 호출을 연결할 수 있습니다. 원하는 속성을 구성한 후에는 build() 메서드를 호출하여 클라이언트를 생성할 수 있습니다.

코드의 고급 구성

다음 코드 조각은 추가 구성 옵션을 보여줍니다.

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

S3Client s3Client = S3Client.builder()
        .region(Region.EU_SOUTH_2)
        .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
        .overrideConfiguration(clientOverrideConfiguration)
        .httpClientBuilder(
                ApacheHttpClient.builder()
                        .maxConnections(100)
                        .connectionTimeout(Duration.ofSeconds(5))
                        .proxyConfiguration(ProxyConfiguration.builder()
                                .endpoint(URI.create("http://proxy:8080"))
                                .build())
        ).build();
```

이전 코드 조각에서는 서비스 클라이언트를 구성하는 몇 가지 진입점을 볼 수 있습니다.

- 모든 서비스 클라이언트에서 공통적인 구성 옵션을 제공하는 **`ClientOverrideConfiguration.Builder` 객체**입니다. 이러한 설정은 HTTP 구현과 무관한 AWS별 동작입니다.
- 별도의 HTTP 클라이언트 빌더 구현을 통한 HTTP 클라이언트 구성 `ApacheHttpClient.Builder`는 예제입니다. 서비스 클라이언트는 구성된 HTTP 클라이언트를 서비스 클라이언트에 연결하는 `httpClientBuilder()` 메서드를 제공합니다.
- 클라이언트 빌더 자체의 메서드(예: `region()`, `credentialsProvider()`)

구성 블록을 사용한 동일한 구성

별도의 객체를 만든 다음 서비스 클라이언트 메서드에 전달하는 대신 AWS SDK for Java 2.x는 Lambda 표현식을 수락하여 이러한 객체를 인라인으로 빌드하는 메서드를 제공합니다. 빌더의 구성 메서드 이름은 동일하지만 서명은 다릅니다. 예:

- `overrideConfiguration(ClientOverrideConfiguration overrideConfiguration)`
- `overrideConfiguration(Consumer<ClientOverrideConfiguration.Builder> overrideConfiguration)`

이 접근 방식을 사용하여 앞서 보여준 S3 클라이언트의 구성은 하나의 코드 블록에서 수행할 수 있습니다.

```
S3Client s3Client = S3Client.builder()
        .region(Region.EU_SOUTH_2)
        .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
        .overrideConfiguration(b -> b
                .apiCallAttemptTimeout(Duration.ofSeconds(1))
                .addMetricPublisher(CloudWatchMetricPublisher.create()))
        .httpClientBuilder(ApacheHttpClient.builder()
                .maxConnections(100)
                .connectionTimeout(Duration.ofSeconds(5))
                .proxyConfiguration(ProxyConfiguration.builder()
                        .endpoint(URI.create("http://proxy:8080"))
                        .build()))
        .build();
```

코드에서 사용할 수 없는 구성 옵션

다음 설정은 SDK의 기본 초기화 프로세스에 영향을 미치므로 다음 구성 설정은 코드 내가 아닌 외부에서만 설정할 수 있습니다.

파일 위치 설정

이러한 설정은 공유 구성 및 자격 증명 파일의 위치를 제어하며 SDK가 파일을 로드한 후에는 프로그래밍 방식으로 재정의할 수 없습니다.

- AWS_CONFIG_FILE(환경 변수)/aws.configFile(JVM 시스템 속성)
- AWS_SHARED_CREDENTIALS_FILE(환경 변수)/aws.sharedCredentialsFile(JVM 시스템 속성)

이러한 설정은 SDK가 구성을 찾는 위치를 결정하므로 SDK가 구성 파일을 로드하기 전에 설정해야 합니다. SDK가 초기화되면 이러한 값을 변경해도 아무런 효과가 없습니다.

인스턴스 메타데이터 서비스 비활성화

- AWS_EC2_METADATA_DISABLED(환경 변수)/aws.disableEc2Metadata(JVM 시스템 속성)

이 설정은 SDK가 EC2 인스턴스 메타데이터 서비스를 사용하려고 시도하는지 여부를 완전히 제어합니다. SDK가 초기화되면 이 설정을 프로그래밍 방식으로 변경할 수 없습니다.

프로파일 선택

- AWS_PROFILE(환경 변수)/aws.profile(JVM 시스템 속성)

이 설정을 통해 공유 구성 및 자격 증명 파일에서 로드할 프로파일을 SDK에 알립니다. 로드된 후에는 이 값을 변경해도 아무런 효과가 없습니다.

컨테이너 자격 증명 경로

- AWS_CONTAINER_CREDENTIALS_RELATIVE_URI
- AWS_CONTAINER_CREDENTIALS_FULL_URI
- AWS_CONTAINER_AUTHORIZATION_TOKEN
- AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE

이러한 환경 변수를 사용하여 SDK에 컨테이너 서비스에서 자격 증명을 가져오는 방법을 알립니다. 서비스 클라이언트 초기화 중에 자격 증명 공급자 체인이 설정되면 이러한 설정을 변경할 수 없습니다.

기본 HTTP 구현 선택

- SYNC_HTTP_SERVICE_IMPL(환경 변수)/software.amazon.awssdk.http.service.impl(JVM 시스템 속성)
- ASYNC_HTTP_SERVICE_IMPL(환경 변수)/software.amazon.awssdk.http.async.service.impl(JVM 시스템 속성)

이러한 전역 설정은 개별 서비스 클라이언트의 코드에서 재정의되지 않는 한 SDK가 모든 서비스 클라이언트에 대해 사용하는 HTTP 클라이언트 구현을 결정합니다. SDK가 HTTP 클라이언트를 초기화하기 전에 이를 설정해야 하며 나중에 변경할 수 없습니다.

AWS SDK for Java 2.x에서 싱글톤 서비스 클라이언트 인스턴스 사용

AWS SDK for Java 2.x의 서비스 클라이언트는 스레드에서 안전합니다. 각 서비스 클라이언트의 인스턴스 하나를 만들고 애플리케이션 전체에서 재사용할 수 있습니다. 이 접근 방식은 성능을 개선하고 리소스를 보다 효율적으로 관리합니다.

싱글톤 서비스 클라이언트의 이점

연결 풀링

서비스 클라이언트는 내부 HTTP 연결 풀을 유지합니다. 이러한 풀을 만들고 삭제하는 데는 비용이 많이 듭니다. 클라이언트를 재사용하면 이러한 풀이 요청 간에 효율적으로 공유됩니다.

초기화 오버헤드 감축

클라이언트를 만들려면 구성을 로드하고, 자격 증명을 설정하고, 내부 구성 요소를 초기화해야 합니다. 싱글톤 인스턴스는 이러한 오버헤드를 없앱니다.

리소스 활용 향상

싱글톤 클라이언트는 많은 클라이언트 인스턴스를 만들 때 발생할 수 있는 리소스 소진을 방지합니다.

싱글톤 서비스 클라이언트 만들기 및 사용

다음 예제 코드에서는 싱글톤 서비스 클라이언트를 만들고 사용하는 방법을 보여줍니다.

```
// Create one instance and use it throughout the application.
public class ServiceClientSource {
    private static final S3Client s3Client = S3Client.create();

    public static S3Client getS3Client() {
        return s3Client;
    }
}
```

각 작업에 대해 새 클라이언트를 만들지 않습니다.

```
// This approach creates unnecessary overhead.
public void badExample() {
    try (S3Client s3 = S3Client.create()) {
        s3.listBuckets();
    }
}
```

중요 고려 사항

- 서비스 클라이언트는 스레드에서 안전합니다. 여러 스레드에서 안전하게 공유할 수 있습니다.
- 애플리케이션이 종료되거나 클라이언트가 더 이상 필요하지 않은 경우에만 클라이언트를 닫습니다. 애플리케이션 수준에서 `client.close()` 리소스 또는 try-with-resources를 사용합니다.
- 리전 또는 자격 증명 등의 다른 구성이 필요한 경우 각 구성에 대해 별도의 싱글톤 인스턴스를 만듭니다.

Spring 등의 종속성 주입 프레임워크를 사용하는 경우 서비스 클라이언트를 싱글톤 bean으로 구성합니다. 이렇게 하면 적절하게 수명 주기를 관리할 수 있습니다.

AWS SDK for Java 2.x에 대한 AWS 리전 설정

SDK 클라이언트는 클라이언트를 만들 때 지정한 특정 AWS 리전의 AWS 서비스에 연결합니다. 이 구성을 사용하면 애플리케이션이 해당 지리적 영역의 AWS 리소스와 상호 작용할 수 있습니다. 리전을 명시적으로 설정하지 않고 서비스 클라이언트를 만들면 SDK는 외부 구성의 기본 리전을 사용합니다.

AWS 리전에 명시적으로 구성

리전을 명시적으로 설정하려면 Regions 클래스에서 정의한 상수를 사용하는 것이 좋습니다. 이 열거형은 공개적으로 사용 가능한 모든 리전의 열거 값입니다.

클래스에서 열거형 리전을 사용하여 클라이언트를 생성하려면 클라이언트 빌더의 `region` 메서드를 사용합니다.

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

사용하려는 리전이 `Region` 클래스의 열거형에 속하지 않는 경우 `of` 메서드를 사용하여 새 리전을 만들 수 있습니다. SDK를 업그레이드하지 않고 새 리전에 액세스 할 수 있는 메서드입니다.

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

> **Note**
>
> 빌더로 클라이언트를 빌드한 후에는 변경할 수 없으며 AWS 리전 리전은 변경할 수 없습니다. 동일한 서비스에 대해 여러 AWS 리전 리전에서 작업해야 하는 경우, 리전당 하나씩 여러 클라이언트를 생성해야 합니다.

SDK가 환경에서 기본 AWS 리전을 자동으로 결정

코드가 Amazon EC2 또는 AWS Lambda에서 실행되는 경우, 코드가 실행되는 것과 동일한 AWS 리전 리전을 사용하도록 클라이언트를 구성할 수 있습니다. 이렇게 하면 실행 중인 환경에서 코드가 분리되므로 낮은 지연 시간과 중복성을 위해 손쉽게 애플리케이션을 여러 AWS 리전에 배포할 수 있습니다.

기본 AWS 리전 공급자 체인을 사용하여 환경에서 리전을 결정하려면 클라이언트 빌더의 `create` 메서드를 사용합니다.

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

다른 방법으로 클라이언트를 구성할 수도 있지만 리전을 설정할 수 없습니다. SDK는 기본 리전 공급자 체인을 사용하여 AWS 리전을 선택합니다.

```
Ec2Client ec2Client = Ec2Client.builder()
        .credentialsProvider(ProfileCredentialsProvider.builder()
                .profileName("my-profile")
                .build())
        .build();
```

`region` 메서드를 사용하여 AWS 리전을 명시적으로 설정하지 않은 경우 SDK는 기본 리전 공급자 체인을 참조하여 사용할 리전을 결정합니다.

기본 AWS 리전 공급자 체인 이해

SDK는 다음 단계를 수행하여 AWS 리전을 찾습니다.

1. 빌더 자체에 대해 `region` 메서드를 사용하여 설정한 명시적 리전을 다른 어떤 것보다 우선합니다.
2. SDK는 JVM 시스템 속성(`aws.region`)을 찾고 발견된 경우 해당 값을 사용합니다.
3. `AWS_REGION` 환경 변수를 확인합니다. 설정한 경우 클라이언트를 구성하는 데 해당 리전이 사용됩니다.

   > **Note**
   >
   > Lambda 컨테이너는 이 환경 변수를 설정합니다.

4. SDK는 AWS 공유 구성 및 자격 증명 파일에서 활성 프로필을 확인합니다. `region` 속성이 있으면 SDK가 이 속성을 사용합니다.

   `default` 프로필은 `AWS_PROFILE` 환경 변수 또는 `aws.profile` JVM 시스템 속성으로 재정의되지 않는 한 활성 프로필입니다. SDK가 동일한 프로필(`default` 프로필 포함)의 두 파일 모두에서 `region` 속성을 찾으면 SDK는 공유 자격 증명 파일의 값을 사용합니다.

5. SDK는 Amazon EC2 인스턴스 메타데이터 서비스(IMDS)를 사용하여 현재 실행 중인 Amazon EC2 인스턴스의 리전을 결정합니다.
   - 보안을 강화하려면 SDK가 IMDS 버전 1을 사용하지 않도록 비활성화해야 합니다. the section called “안전” 섹션에 설명된 것과 동일한 설정을 사용하여 버전 1을 비활성화합니다.
6. 이때까지도 SDK에서 여전히 리전을 찾지 못한 경우 클라이언트 생성이 실패하고 예외가 발생합니다.

AWS 애플리케이션을 개발할 때 일반적인 접근 방법은 공유 구성 파일을 사용하여 로컬 개발용으로 리전을 설정하고, 기본 리전 공급자 체인에 의존하여 애플리케이션이 AWS 인프라에서 실행할 때 리전을 결정하는 것입니다. 이렇게 하면 클라이언트 생성 작업이 크게 간소화되며 애플리케이션을 이식 가능한 형태로 유지됩니다.

리전에서 서비스를 사용할 수 있는지 확인합니다.

리전에서 특정 AWS 서비스를 사용할 수 있는지 확인하려면 서비스 클라이언트에서 `serviceMetadata` 메서드를 사용합니다.

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

이전 코드 조각은 DynamoDB 서비스가 있는 긴 AWS 리전 코드 목록을 출력합니다.

```
af-south-1
ap-east-1
ap-northeast-1
ap-northeast-2
ap-northeast-3
ap-south-1
ap-south-2
ap-southeast-1
...
```

코드를 사용하여 서비스 클라이언트가 사용해야 하는 리전의 리전 클래스 열거를 조회할 수 있습니다.

예를 들어 코드(`ap-northeast-2`)를 사용하여 리전에서 DynamoDB로 작업하려면 최소한 다음 구성으로 DynamoDB 클라이언트를 만듭니다.

```
DynamoDbClient ddb = DynamoDbClient.builder()
    .region(Region.AP_NORTHEAST_2)
    .build();
```

특정 엔드포인트 선택

특정 상황(예: 기능이 일반 사용 가능성으로 전환되기 전에 서비스의 미리 보기 기능을 테스트하는 경우)에서는 리전에서 특정 엔드포인트를 지정해야 할 수도 있습니다. 이러한 상황에서는 `endpointOverride` 메서드를 호출하여 서비스 클라이언트를 구성할 수 있습니다.

예를 들어 특정 엔드포인트가 있는 유럽(아일랜드) 리전을 사용하도록 Amazon EC2 클라이언트를 구성하려면 다음 코드를 사용하세요.

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

모든 AWS 서비스에 대한 리전 및 해당 엔드포인트 현재 목록은 리전 및 엔드포인트를 참조하세요.

에서 자격 증명 공급자 사용 AWS SDK for Java 2.x

에서 자격 증명 공급자의 역할은 SDK의 AWS 서비스 클라이언트 AWS SDK for Java 2.x 에 자격 증명을 소싱하고 제공하는 것입니다. SDK는 획득한 자격 증명을 사용하여 각 요청에 암호화 방식으로 서명하여 서비스를 인증합니다. 자격 증명은 일반적으로 액세스 키(액세스 키 ID)와 시크릿 액세스 키로 구성됩니다.

SSO 토큰 공급자 구성을 설정하거나 IAM(AWS Identity and Access Management) 역할을 예로 들도록 런타임을 구성할 때 사용되는 임시 자격 증명을 사용하면 액세스 키에 세션 토큰이 추가되어 AWS 리소스에 대한 시간 제한 액세스를 제공합니다.

이 주제에서는 자격 증명에 액세스하기 위해 SDK를 활성화하는 여러 가지 방법에 대해 설명합니다.

주제

- AWS SDK for Java 2.x를 사용한 대화형 개발 작업용 액세스 자격 증명
- AWS SDK for Java 2.x의 기본 자격 증명 공급자 체인
- 의 자격 증명 캐싱 AWS SDK for Java 2.x
- 에서 특정 자격 증명 공급자 지정 AWS SDK for Java 2.x
- 에서 AWS 공유 구성 프로필 사용 AWS SDK for Java 2.x
- AWS SDK for Java 2.x를 사용한 외부 프로세스에서 자격 증명 로드
- AWS SDK for Java 2.x를 사용하여 코드에 자격 증명 제공
- SDK for Java 2.x를 사용하여 Amazon EC2에서 IAM 역할 자격 증명 읽기

AWS SDK for Java 2.x를 사용한 대화형 개발 작업용 액세스 자격 증명

보안을 강화하려면 수명이 긴 자격 증명 대신 임시 자격 증명을 사용하도록 Java용 SDK를 구성하는 것이 AWS 좋습니다. 임시 자격 증명은 액세스 키(액세스 키 ID 및 시크릿 액세스 키)와 세션 토큰으로 구성됩니다.

임시 자격 증명으로 작업할 수 있는 몇 가지 접근 방식을 사용할 수 있습니다. 사용하는 접근 방식과 SDK에 제공하는 구성은 사용 사례에 따라 다릅니다.

Java SDK로 대화형 개발 작업을 수행할 때는 AWS 콘솔 로그인 자격 증명을 사용하는 것이 좋습니다.

콘솔 로그인 자격 증명 사용

기존 AWS Management Console 로그인 자격 증명을 사용하여 AWS 서비스에 프로그래밍 방식으로 액세스할 수 있습니다. 브라우저 기반 인증 흐름 후는 AWS CLI 및 Java 2.x용 SDK와 같은 로컬 개발 도구에서 작동하는 임시 자격 증명을 AWS 생성합니다.

이 프로세스를 사용하면 초기 계정 설정 중에 생성된 루트 자격 증명, IAM 사용자 또는 자격 증명 공급자의 페더레이션 자격 증명을 사용하여 인증할 수 있으며,가 AWS CLI 자동으로 임시 자격 증명을 관리합니다. 이 접근 방식은 장기 자격 증명을 로컬에 저장할 필요가 없으므로 보안이 강화됩니다.

`aws login` 명령을 실행하면 활성 콘솔 세션에서 선택하거나 브라우저 기반 인증 흐름을 통해 로그인할 수 있으며, 그러면 임시 자격 증명이 자동으로 생성됩니다. Java 2.x용 SDK는 최대 12시간 동안 이러한 자격 증명을 자동으로 새로 고칩니다.

> **Important**
>
> 모든 프로젝트에 작동하는 공유 구성 파일에서 설정한 구성 외에도 각 개별 Java 프로젝트에는 Maven `pom.xml` 파일에 다음과 같은 종속성이 필요합니다.
>
> ```
> <dependency>
>     <groupId>software.amazon.awssdk</groupId>
>     <artifactId>signin</artifactId>
> </dependency>
> ```
>
> `signin` 종속성은 SDK for Java 2.x가 콘솔 로그인 자격 증명에 액세스하고 사용할 수 있도록 하는 코드를 제공합니다.

사전 시퀀스, 로그인 및 로그아웃에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서의 콘솔 자격 증명을 사용하여 AWS 로컬 개발을 위한 로그인을 참조하세요.

Single-sign-on 접근 방식

Java SDK로 대화형 개발 작업을 수행할 때 Single Sign-On 접근 방식을 사용할 수도 있습니다. 이 접근 방식을 사용하려면 다음 설정이 필요합니다.

- IAM Identity Center를 통해 설정
- AWS 공유 구성 파일에서 프로필 구성
- 를 사용하고 명령을 AWS CLI 실행하여 로그인 및 활성 세션 생성 ???

IAM Identity Center 구성

이 안내서의 ???에 설명된 대로 IAM Identity Center Single Sign-On 액세스를 사용하도록 SDK를 구성하면 SDK는 임시 자격 증명을 사용합니다.

SDK는 IAM Identity Center 액세스 토큰을 사용하여 `config` 파일의 `sso_role_name` 설정으로 구성된 IAM 역할에 대한 액세스 권한을 얻습니다. SDK는 이 IAM 역할을 맡고 AWS 서비스 요청에 서명할 임시 자격 증명을 검색합니다.

SDK가 구성에서 임시 자격 증명을 가져오는 방법에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서의 IAM Identity Center 인증 이해 단원을 참조하세요.

> **Important**
>
> 모든 프로젝트에서 작동하는 공유 `config` 파일에 설정한 구성 외에도 각 개별 Java 프로젝트에는 Maven `pom.xml` 파일에 다음과 같은 종속성이 필요합니다.
>
> ```
> <dependency>
>     <groupId>software.amazon.awssdk</groupId>
>     <artifactId>sso</artifactId>
> </dependency>
> <dependency>
>     <groupId>software.amazon.awssdk</groupId>
>     <artifactId>ssooidc</artifactId>
> </dependency>
> ```
>
> `sso` 및 `ssooidc` 종속성은 SDK for Java 2.x가 임시 자격 증명에 액세스할 수 있도록 하는 코드를 제공합니다.

AWS 액세스 포털에서 임시 자격 증명 검색

IAM Identity Center Single Sign-On 구성의 대안으로 AWS 액세스 포털에서 사용할 수 있는 임시 자격 증명을 복사하고 사용할 수 있습니다. 프로파일에서 임시 보안 인증을 사용하거나 이를 시스템 속성 및 환경 변수의 값으로 사용할 수 있습니다.

임시 자격 증명 파일을 위한 로컬 자격 증명 파일 설정

1. 공유 자격 증명 파일 생성
2. 자격 증명 파일에서 작업 중인 임시 자격 증명을 붙여넣을 때까지 다음 자리 표시자 텍스트를 붙여넣습니다.

```
[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. 파일을 저장합니다. 이제 파일 `~/.aws/credentials`이 로컬 개발 시스템에 존재해야 합니다. 이 파일에는 이름이 지정된 특정 프로필이 지정되지 않은 경우 Java용 SDK에서 사용하는 [기본] 프로필이 들어 있습니다.
4. AWS 액세스 포털에 로그인
5. 수동 자격 증명 새로 고침 제목 아래의 다음 지침에 따라 AWS 액세스 포털에서 IAM 역할 자격 증명을 복사합니다.

   a. 링크된 설명의 2단계에서 개발 요구 사항에 따라 액세스 권한을 부여하는 IAM 역할 이름에 대해 `Access keys`를 선택합니다. 이 역할은 일반적으로 PowerUserAccess 또는 Developer와 같은 이름을 갖습니다.

   b. 모달 대화 상자에서 운영 체제를 선택하고 프로파일 추가의 콘텐츠를 AWS 자격 증명 파일에 복사합니다.
6. 복사한 자격 증명을 로컬 `credentials` 파일에 붙여넣고 붙여넣은 프로필 이름을 모두 제거합니다. 파일은 다음과 유사해야 합니다.

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

7. `credentials` 파일을 저장합니다.

SDK는 서비스 클라이언트를 생성할 때 이러한 임시 자격 증명에 액세스하여 각 요청에 사용합니다. 5a단계에서 선택한 IAM 역할 설정에 따라 임시 보안 인증의 유효 기간이 결정됩니다. 최대 유효 기간은 12시간입니다.

임시 보안 인증이 만료되면 4~7단계를 반복합니다.

AWS SDK for Java 2.x의 기본 자격 증명 공급자 체인

의 기본 자격 증명 공급자 체인은 미리 정의된 위치 시퀀스에서 AWS 자격 증명을 AWS SDK for Java 2.x 자동으로 검색하므로 애플리케이션이 자격 증명 소스를 명시적으로 지정 AWS 서비스 하지 않고 로 인증할 수 있습니다.

기본 자격 증명 공급자 체인은 DefaultCredentialsProvider 클래스에 의해 구현됩니다. 다양한 위치에서 구성을 확인하는 다른 자격 증명 공급자 구현에 순차적으로 위임합니다. 모든 필수 구성 요소를 찾을 수 있는 첫 번째 자격 증명 공급자가 체인을 종료합니다.

기본 자격 증명 공급자 체인을 사용하여 임시 자격 증명을 제공하려면 서비스 클라이언트 빌더를 만들되 자격 증명 공급자는 지정하지 않습니다. 다음 코드 조각은 기본 자격 증명 공급자 체인을 사용하여 구성 설정을 찾고 검색하는 DynamoDbClient를 만듭니다.

```
// Any external Region configuration is overridden.
// The SDK uses the default credentials provider chain because no specific credentials
 provider is specified.
Region region = Region.US_WEST_2;
DynamoDbClient ddb =
    DynamoDbClient.builder()
                  .region(region)
                  .build();
```

자격 증명 설정 검색 순서

Java 2.x용 SDK의 기본 자격 증명 공급자 체인은 사전 정의된 시퀀스를 사용하여 사용자 환경의 구성을 검색합니다.

1. Java 시스템 속성
   - SDK는 SystemPropertyCredentialsProvider 클래스를 사용하여, aws.accessKeyId, aws.secretAccessKey, 및 aws.sessionToken Java 시스템 속성에서 임시 자격 증명을 로드합니다.

     > **Note**
     >
     > Java 시스템 속성을 설정하는 방법에 대한 자세한 내용은 공식 Java Tutorials 웹 사이트의 시스템 속성 자습서를 참조하세요.

2. 환경 변수

- SDK는 EnvironmentVariableCredentialsProvider 클래스를 사용하여 `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` 및 `AWS_SESSION_TOKEN` 환경 변수에서 임시 자격 증명을 로드합니다.

3. 웹 ID 토큰 및 IAM 역할 ARN
   - SDK는 WebIdentityTokenFileCredentialsProvider 클래스를 사용하여 웹 ID 토큰으로 역할을 수임해 자격 증명을 로드합니다.
   - 자격 증명 공급자는 다음 환경 변수 또는 JVM 시스템 속성을 찾습니다.
     - `AWS_WEB_IDENTITY_TOKEN_FILE or aws.webIdentityTokenFile`
     - `AWS_ROLE_ARN` 또는 `aws.roleArn`
     - `AWS_ROLE_SESSION_NAME` 또는 `aws.roleSessionName`(선택 사항)
   - SDK는 값을 획득하면 AWS Security Token Service (STS)를 호출하고 반환하는 임시 자격 증명을 사용하여 요청에 서명합니다.
   - Amazon Elastic Kubernetes Service(EKS)와 같은 런타임 환경은 웹 자격 증명 토큰을 AWS SDKs에 자동으로 제공하여 애플리케이션이 임시 AWS 자격 증명을 얻을 수 있도록 합니다.
4. 공유 `credentials` 및 `config` 파일
   - SDK는 ProfileCredentialsProvider를 사용하여 IAM Identity Center 싱글 사인온 설정 또는 `[default]` 프로필의 임시 자격 증명을 공유 `credentials` 및 `config` 파일에 로드합니다.

     AWS SDKs 및 도구 참조 안내서에는 SDK for Java가 IAM Identity Center Single Sign-On 토큰과 함께 작동하여 SDK가 호출하는 데 사용하는 임시 자격 증명을 가져오는 방법에 대한 자세한 정보가 나와 있습니다 AWS 서비스.

     > **Note**
     >
     > `credentials` 및 `config` 파일은 AWS SDKs 및 도구에서 공유됩니다. 자세한 정보는 AWS SDK 및 도구 참조 가이드의 .aws/credentials 및 .aws/config 파일을 참조하세요.

   - 공유 `credentials` 및 `config` 파일의 프로파일에는 많은 다양한 설정 세트가 포함될 수 있으므로 `ProfileCredentialsProvider`는 일련의 다른 공급자에 위임하여 `[default]` 프로파일에서 설정을 찾습니다.
     - 웹 ID 토큰 자격 증명(`WebIdentityTokenCredentialsProvider` 클래스): 프로파일에 `role_arn`, `web_identity_token_file`이 포함된 경우
     - SSO 자격 증명(`SsoCredentialsProvider` 클래스): 프로파일에 `sso_role_name`, `sso_account_id`와 같은 SSO 관련 속성이 포함된 경우

- 소스 프로파일이 있는 역할 기반 자격 증명(StsAssumeRoleCredentialsProvider 클래스): 프로파일에 role_arn, source_profile이 포함된 경우
- 자격 증명 소스가 있는 역할 기반 자격 증명(StsAssumeRoleWithSourceCredentialsProvider 클래스): 프로파일에 role_arn, credential_source가 포함된 경우
  - credential_source = Environment인 경우: SystemPropertyCredentialsProvider, EnvironmentVariableCredentialsProvider의 체인 사용
  - credential_source = Ec2InstanceMetadata인 경우: InstanceProfileCredentialsProvider 사용
  - credential_source = EcsContainer인 경우: ContainerCredentialsProvider 사용
- 콘솔 로그인 자격 증명(클래스 LoginCredentialsProvider): 프로파일에가 포함된 경우 login_session
- 프로세스 자격 증명(ProcessCredentialsProvider 클래스): 프로파일에 credential_process가 포함된 경우
- 세션 자격 증명(StaticSessionCredentialsProvider 클래스): 프로파일에 aws_access_key_id, aws_secret_access_key, aws_session_token이 포함된 경우
- 기본 자격 증명(StaticCredentialsProvider 클래스): 프로파일에 aws_access_key_id 및 aws_secret_access_key가 포함된 경우

5. Amazon ECS 컨테이너 자격 증명
   - SDK는 ContainerCredentialsProvider 클래스를 사용하여 다음 환경 변수로 임시 자격 증명을 로드합니다.
     a. AWS_CONTAINER_CREDENTIALS_RELATIVE_URI 또는 AWS_CONTAINER_CREDENTIALS_FULL_URI
     b. AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE 또는 AWS_CONTAINER_AUTHORIZATION_TOKEN

   ECS 컨테이너 에이전트는 ECS 자격 증명 엔드포인트를 가리키는 AWS_CONTAINER_CREDENTIALS_RELATIVE_URI 환경 변수를 자동으로 설정합니다. 일반적으로 다른 환경 변수는 표준 ECS 자격 증명 엔드포인트가 사용되지 않는 특정 시나리오에서 설정됩니다.
6. Amazon EC2 인스턴스 IAM 역할 제공 자격 증명
   - SDK는 InstanceProfileCredentialsProvider 클래스를 사용하여 Amazon EC2 메타데이터 서비스에서 임시 자격 증명을 로드합니다.

7. SDK가 위에 나열된 모든 단계를 통해 필요한 구성 설정을 찾을 수 없는 경우 다음과 비슷한 출력이 포함된 예외가 발생합니다.

```
software.amazon.awssdk.core.exception.SdkClientException: Unable to load credentials
 from any of the providers
in the chain
 AwsCredentialsProviderChain(credentialsProviders=[SystemPropertyCredentialsProvider(),
EnvironmentVariableCredentialsProvider(), WebIdentityTokenCredentialsProvider(),
 ProfileCredentialsProvider(),
ContainerCredentialsProvider(), InstanceProfileCredentialsProvider()])
```

코드에서 **DefaultCredentialsProvider** 사용

코드에서 기본 자격 증명 공급자 체인을 명시적으로 사용할 수 있습니다. 이는 SDK가 기본적으로 `DefaultCredentialsProvider`를 사용하기 때문에 자격 증명 공급자를 전혀 지정하지 않는 것과 기능적으로 동일합니다. 그러나 명시적으로 사용하면 코드를 더 읽기 쉽고 자체적으로 문서화할 수 있습니다. 기본 자격 증명 체인을 사용하려는 의도를 명확하게 보여줍니다.

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;

public class ExplicitDefaultCredentialsExample {
    public static void main(String[] args) {
        // Explicitly create the DefaultCredentialsProvider.
        DefaultCredentialsProvider defaultCredentialsProvider =
 DefaultCredentialsProvider
                                                                .builder().build();

        // Use it with any service client.
        S3Client s3Client = S3Client.builder()
            .region(Region.US_WEST_2)
            .credentialsProvider(defaultCredentialsProvider)
            .build();

        // Now you can use the client with the default credentials chain.
        s3Client.listBuckets();
    }
}
```

기본 자격 증명 공급자를 구축할 때 추가 구성을 제공할 수 있습니다.

```
DefaultCredentialsProvider customizedProvider = DefaultCredentialsProvider.builder()
    .profileName("custom-profile")  // Use a specific profile if the chain gets to the
 `ProfileCredentialsProvider` stage.
    .asyncCredentialUpdateEnabled(true)  // Enable async credential updates.
    .build();
```

이 접근 방식을 사용하면 기본 자격 증명 체인의 편의성을 유지하면서 더 많은 제어 권한을 얻을 수 있습니다.

의 자격 증명 캐싱 AWS SDK for Java 2.x

는 자격 증명 캐싱을 AWS SDK for Java 2.x 구현하여 성능을 개선하고 자격 증명 소스에 대한 호출을 줄입니다. 이 섹션에서는 자격 증명 캐싱의 작동 방식과 애플리케이션에 맞게 자격 증명을 구성하는 방법을 설명합니다.

자격 증명 공급자 캐싱 이해

SDK for Java 2.x의 자격 증명 공급자는 다양한 캐싱 전략을 사용합니다.

- 내부 자격 증명 캐싱: 많은 공급자가 검색한 자격 증명을 캐시합니다.
- 자동 새로 고침: 캐시된 자격 증명이 있는 공급자는 새로 고침 메커니즘을 구현합니다.

내부 자격 증명 캐싱이 있는 공급자

다음 자격 증명 공급자는 새 인스턴스를 만들 때도 자격 증명을 내부적으로 캐시합니다.

- 인스턴스 프로파일 자격 증명 공급자: Amazon EC2 메타데이터 서비스에서 자격 증명을 캐시합니다.
- 컨테이너 자격 증명 공급자: 컨테이너 메타데이터 엔드포인트에서 자격 증명을 캐시합니다.
- STS 기반 공급자: AWS Security Token Service (STS)에서 임시 자격 증명을 캐시합니다.
- 웹 ID 토큰 공급자: 웹 ID 토큰에서 얻은 자격 증명을 캐시합니다.
- 프로세스 자격 증명 공급자: 외부 프로세스에서 자격 증명을 캐시합니다.

내부 캐싱이 없는 공급자

다음 공급자는 내부 캐싱을 구현하지 않습니다.

- 환경 변수 자격 증명 공급자

- 시스템 속성 자격 증명 공급자
- 정적 자격 증명 공급자

자격 증명 캐싱 구성

자격 증명 공급자를 구축할 때 캐싱 동작을 사용자 지정할 수 있습니다.

기한 경과 시간

자격 증명이 오래된 것으로 간주되어 새로 고침이 필요한 경우를 제어합니다.

```
.staleTime(Duration.ofMinutes(2))  // Consider stale 2 minutes before expiration.
```

미리 가져오기 시간

만료되기 전에 자격 증명 새로 고침을 시작할 시기를 결정합니다.

```
.prefetchTime(Duration.ofMinutes(10))  // Start refresh 10 minutes before expiration.
```

비동기식 업데이트

백그라운드 자격 증명 새로 고침을 사용합니다.

```
.asyncCredentialUpdateEnabled(true)  // Refresh credentials in background thread.
```

세션 지속 시간

STS 기반 공급자의 경우는 임시 자격 증명이 유효한 기간을 제어합니다.

```
.refreshRequest(r -> r.durationSeconds(3600))  // 1 hour session.
```

자격 증명 구성 캐싱 예제

자격 증명 공급자 구현을 위한 캐싱을 구성하는 예로 SDK가 백그라운드 스레드를 사용하여 자격 증명이 만료되기 전에 미리 가져오도록(사전 검색) 할 수 있습니다. 이렇게 하면 새 자격 증명을 검색하는 차단 직접 호출을 방지할 수 있습니다.

다음은 빌더에서 `asyncCredentialUpdateEnabled` 속성을 `true`로 설정하여 백그라운드 스레드를 사용해 자격 증명을 미리 가져오는 `StsAssumeRoleCredentialsProvider`를 만드는 예제입니다.

```
StsAssumeRoleCredentialsProvider provider = StsAssumeRoleCredentialsProvider.builder()
    .refreshRequest(r -> r
        .roleArn("arn:aws:iam::111122223333:role/example-role")
        .roleSessionName("example-session")
        .durationSeconds(3600))  // 1 hour session
    .staleTime(Duration.ofMinutes(5))  // Consider stale 5 minutes before expiration
    .prefetchTime(Duration.ofMinutes(10))  // Start refresh 10 minutes before
 expiration
    .asyncCredentialUpdateEnabled(true)  // Refresh in background
    .build();

S3Client s3 = S3Client.builder()
    .credentialsProvider(provider)
    .build();
```

에서 작업을 `s3Client` 처음 호출하면 `AssumeRoleRequest`가 AWS Security Token Service (STS)로 전송됩니다. STS는 15분(900초) 동안 유효한 임시 자격 증명을 반환합니다. `s3Client` 인스턴스는 15분이 경과하기 전에 새로 고침될 때까지 캐시된 자격 증명을 사용합니다. 기본적으로 SDK는 현재 세션의 만료 시간 5분~1분 전에 새 세션에 대한 새 자격 증명을 검색하려고 시도합니다. 사전 가져오기 기간에는 `prefetchTime` 및 `staleTime` 속성을 사용하여 구성할 수 있습니다.

다음과 같은 세션 기반 자격 증명 공급자를 유사하게 구성할 수 있습니다.

- `StsWebIdentityTokenFileCredentialsProvider`
- `StsGetSessionTokenCredentialsProvider`
- `StsGetFederationTokenCredentialsProvider`
- `StsAssumeRoleWithWebIdentityCredentialsProvider`
- `StsAssumeRoleWithSamlCredentialsProvider`
- `StsAssumeRoleCredentialsProvider`
- `DefaultCredentialsProvider`(세션을 사용하는 자격 증명 공급자에 위임하는 경우)
- `ProcessCredentialsProvider`
- `WebIdentityTokenFileCredentialsProvider`
- `ContainerCredentialsProvider`
- `InstanceProfileCredentialsProvider`

자격 증명 캐싱을 이해하면 작업 시 애플리케이션의 성능과 안정성을 최적화하는 데 도움이 됩니다 AWS 서비스.

에서 특정 자격 증명 공급자 지정 AWS SDK for Java 2.x

기본 자격 증명 공급자 체인은 많은 시나리오에서 편리하지만 자격 증명 공급자를 명시적으로 지정하면 인증 동작, 성능 및 보안을 더 잘 제어할 수 있습니다.

자격 증명 공급자를 지정하려는 이유는 다음과 같습니다.

- 기본 공급자 체인은 여러 소스를 순차적으로 확인하여 지연 시간을 추가할 수 있습니다.

```
// The default provider chain checks might check multiple sources until it finds
// sufficient configuration.
S3Client s3Client = S3Client.builder().build();

// You can specify exactly where to look.
S3Client optimizedClient = S3Client.builder()
    .credentialsProvider(InstanceProfileCredentialsProvider.create())
    .build();
```

- 자격 증명 구성에 액세스하려면 비표준 위치를 사용해야 합니다.

```
// Use configuration from a custom file location.
S3Client s3Client = S3Client.builder()
    .credentialsProvider(ProfileCredentialsProvider.builder()
            .profileFile(ProfileFile.builder()
                    .content(Paths.get("/custom/path/to/configuration/file"))
                    .type(ProfileFile.Type.CONFIGURATION) // Expects all non-default
 profiles to be prefixed with "profile".
                    .build())
            .profileName("custom")
            .build())
    .build();
```

- 다양한 서비스 클라이언트에서 다른 자격 증명을 사용합니다. 예를 들어 애플리케이션이 여러 AWS 계정에 액세스해야 하거나 다른 서비스에 대해 다른 권한을 사용해야 하는 경우:

```
// S3 client using one set of credentials.
S3Client s3Client = S3Client.builder()
    .credentialsProvider(ProfileCredentialsProvider.create("s3-readonly"))
    .build();

// DynamoDB client using different credentials.
DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
    .credentialsProvider(ProfileCredentialsProvider.create("dynamodb-admin"))
```

```
    .build();
```

- 자격 증명 새로 고침 동작 제어:

```
// Create a provider with custom refresh behavior.
StsAssumeRoleCredentialsProvider customRefreshProvider =
    StsAssumeRoleCredentialsProvider.builder()
        .refreshRequest(AssumeRoleRequest.builder()
            .roleArn("arn:aws:iam::123456789012:role/my-role")
            .roleSessionName("custom-session")
            .build())
        .stsClient(StsClient.create())
        .asyncCredentialUpdateEnabled(true) // Use a background thread to prefetch
 credentials.
        .build();

S3Client s3Client = S3Client.builder()
    .credentialsProvider(customRefreshProvider)
    .build();
```

에서 AWS 공유 구성 프로필 사용 AWS SDK for Java 2.x

공유 `config` 및 `credentials` 파일을 사용하여 여러 프로필을 설정할 수 있습니다. 이렇게 하면 애플리케이션에서 여러 자격 증명 구성 세트를 사용할 수 있습니다. `[default]` 프로필은 앞서 언급한 바 있습니다. SDK는 ProfileCredentialsProvider 클래스를 사용하여 공유 `credentials` 파일에 정의된 프로필에서 설정을 로드합니다.

다음 코드 조각은 이름이 `my_profile`로 지정된 프로필의 일부로 정의된 설정을 사용하는 서비스 클라이언트를 빌드하는 방법을 보여줍니다.

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

다른 프로필을 기본값으로 설정

`[default]` 프로필 이외의 프로필을 애플리케이션의 기본 프로필로 설정하려면 `AWS_PROFILE` 환경 변수를 사용자 지정 프로필 이름으로 설정합니다.

Linux, macOS 또는 Unix에서 이러한 변수를 설정하려면 `export`를 사용합니다.

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

Windows에서 이러한 변수를 설정하려면 `set`을 사용합니다.

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

또는 `aws.profile` Java 시스템 속성을 프로필 이름으로 설정합니다.

프로필 자격 증명 다시 로드

빌더에 프로필 자격 증명을 다시 로드하는 `profileFile()` 메서드가 있는 모든 자격 증명 공급자를 구성할 수 있습니다. 이러한 자격 증명 프로필 클래스는 `ProfileCredentialsProvider`, `DefaultCredentialsProvider`, `InstanceProfileCredentialsProvider`, 및 `ProfileTokenProvider.`입니다.

> **Note**
>
> 프로필 자격 증명 재로드는 프로필 파일의 다음 설정(`aws_access_key_id`, `aws_secret_access_key` 및 `aws_session_token`)에서만 작동합니다. `region`, `sso_session`, `sso_account_id`, 및 `source_profile` 등의 설정은 무시됩니다.

지원되는 자격 증명 공급자가 프로필 설정을 다시 로드하도록 구성하려면 `profileFile()` 빌더 메서드에 `ProfileFileSupplier` 인스턴스를 제공하세요. 다음 코드 예제는 `[default]` 프로필에서 자격 증명 설정을 다시 로드하는 `ProfileCredentialsProvider`를 보여줍니다.

```
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().
*/
```

ProfileCredentialsProvider.resolveCredentials()가 호출되면 Java용 SDK가 설정을 다시 로드합니다. ProfileFileSupplier.defaultSupplier()는 SDK에서 제공하는 ProfileFileSupplier의 여러 편의 구현 중 하나입니다. 사용 사례에 필요한 경우 자체 구현을 제공할 수 있습니다.

다음 예제는 ProfileFileSupplier.reloadWhenModified() 편의 메서드를 사용하는 방법을 보여줍니다. reloadWhenModified()는 Path 매개 변수를 사용하여 표준 ~/.aws/credentials(또는 config) 위치가 아닌 구성의 소스 파일을 유연하게 지정할 수 있습니다.

SDK에서 파일 내용이 수정되었다고 판단하는 경우에만 resolveCredentials()가 호출될 때 설정이 다시 로드됩니다.

```
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.
*/
```

ProfileFileSupplier.aggregate() 메서드는 여러 구성 파일의 내용을 병합합니다. 파일을 resolveCredentials()를 호출할 때마다 다시 로드할지 아니면 파일을 처음 읽을 때 파일 설정을 고정할지를 결정합니다.

다음 예제는 프로필 설정이 포함된 두 파일의 설정을 병합하는 DefaultCredentialsProvider를 보여줍니다. SDK는 resolveCredentials()가 호출되고 설정이 변경될 때마다 credentialsFilePath 변수가 가리키는 파일에 설정을 다시 로드합니다. profileFile 객체의 설정은 동일하게 유지됩니다.

```
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.
*/
```

AWS SDK for Java 2.x를 사용한 외부 프로세스에서 자격 증명 로드

> **Warning**
>
> 다음은 외부 프로세스에서 자격 증명을 소싱하는 방법을 설명합니다. 이는 잠재적으로 위험할 수 있으므로 주의해서 진행하세요. 가능하면 다른 자격 증명 공급자를 사용하는 것이 좋습니다. 이 옵션을 사용하는 경우 운영 체제의 보안 모범 사례를 사용하여 가능하면 `config` 파일을 잠그도록 해야 합니다.
>
> 사용자 지정 자격 증명 도구가 `StdErr`에 비밀 정보를 기록하지 않도록 하세요. SDKs 이러한 정보를 AWS CLI 캡처하고 로깅하여 권한이 없는 사용자에게 노출될 수 있습니다.

Java 2.x용 SDK를 사용하면 외부 프로세스로부터 사용자 지정 사용 사례에 대한 임시 자격 증명을 얻을 수 있습니다. 이 기능을 구성하는 방법에는 두 가지가 있습니다.

`credential_process` 설정 사용

임시 자격 증명을 제공하는 방법이 있는 경우 `credential_process` 설정을 프로필 정의의 일부로 `config` 파일에 추가하여 통합할 수 있습니다. 지정하는 값은 명령 파일의 전체 경로를 사용해야 합니다. 파일 경로에 공백이 있는 경우 따옴표로 감싸야 합니다.

SDK가 명령을 그대로 정확하게 호출한 후 `stdout`에서 JSON 데이터를 읽어옵니다.

다음 예에서는 공백이 없는 파일 경로와 공백이 있는 파일 경로에 이 설정을 사용하는 방법을 보여줍니다.

Linux/macOS

파일 경로에 공백이 없음

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

파일 경로에 공백

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

Windows

파일 경로에 공백이 없음

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

파일 경로에 공백

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

다음 코드 조각은 이름이 `process-credential-profile`로 지정된 프로필의 일부로 정의된 임시 자격 증명을 사용하는 서비스 클라이언트를 빌드하는 방법을 보여줍니다.

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

외부 프로세스를 임시 자격 증명의 소스로 사용하는 방법에 대한 자세한 내용은 SDK 및 도구 참조 안내서의 프로세스 자격 증명 섹션을 참조하세요. AWS SDKs

`ProcessCredentialsProvider` 사용

`config` 파일의 설정을 사용하는 대신 SDK의 `ProcessCredentialsProvider`를 사용하여 Java로 임시 자격 증명을 로드할 수 있습니다.

다음 예제는 `ProcessCredentialsProvider`를 사용하여 외부 프로세스를 지정하고 임시 자격 증명을 사용하는 서비스 클라이언트를 구성하는 다양한 버전의 방법을 보여줍니다.

Linux/macOS

파일 경로에 공백이 없음

```
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();
```

파일 경로에 공백

```
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

파일 경로에 공백이 없음

```
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();
```

파일 경로에 공백

```
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();
```

인증 시 IAM Roles Anywhere 사용

IAM Roles Anywhere는 외부에서 실행되는 워크로드에 대한 임시 AWS 자격 증명을 얻을 수 AWS 서비스 있는 입니다 AWS. 온프레미스 또는 기타 클라우드 환경에서 AWS 리소스에 안전하게 액세스할 수 있습니다.

IAM Roles Anywhere로 요청을 인증하려면 먼저 필요한 정보를 수집하고 자격 증명 도우미 도구를 다운로드해야 합니다. IAM Roles Anywhere 사용 설명서의 시작하기 설명에 따라 필요한 아티팩트를 만들 수 있습니다.

SDK for Java에는 IAM Roles Anywhere에서 임시 자격 증명을 검색할 수 있는 전용 자격 증명 공급자가 없지만, 자격 증명 도우미 도구를 옵션 중 하나와 함께 사용하여 외부 프로세스에서 자격 증명을 검색할 수 있습니다.

프로파일에서 **credential_process** 설정 사용

공유 구성 파일의 다음 코드 조각은 credential_process 설정을 roles_anywhere 사용하는 AWS 라는 프로필을 보여줍니다.

```
[profile roles_anywhere]
credential_process = ./aws_signing_helper credential-process \
  --certificate /path/to/certificate \
  --private-key /path/to/private-key \
  --trust-anchor-arn arn:aws:rolesanywhere:region:account:trust-anchor/TA_ID \
  --profile-arn arn:aws:rolesanywhere:region:account:profile/PROFILE_ID \
  --role-arn arn:aws:iam::account:role/role-name-with-path
```

모든 아티팩트를 조합한 후 빨간색으로 표시된 텍스트를 값으로 대체해야 합니다. 설정의 첫 번째 요소인 aws_signing_helper는 자격 증명 도우미 도구의 실행 파일이고 credential-process는 명령입니다.

다음 코드와 같이 roles_anywhere 프로파일을 사용하도록 서비스 클라이언트를 구성하면 SDK는 임시 자격 증명을 캐시하고 만료되기 전에 새로 고침합니다.

```
S3Client s3Client = S3Client.builder()
    .credentialsProvider(ProfileCredentialsProvider.builder()
        .profileName("roles_anywhere").build())
    .build();
```

ProcessCredentialsProvider 구성

다음 그림과 같이 프로파일 설정을 사용하는 대신 ProcessCredentialsProvider에서 코드 전용 접근 방식을 사용할 수 있습니다.

```
ProcessCredentialsProvider processCredentialsProvider =
 ProcessCredentialsProvider.builder()
    .command("""
            ./aws_signing_helper credential-process \
            --certificate /path/to/certificate \
            --private-key /path/to/private-key \
            --trust-anchor-arn arn:aws:rolesanywhere:region:account:trust-anchor/TA_ID
 \
```

```
            --profile-arn arn:aws:rolesanywhere:region:account:profile/PROFILE_ID \
            --role-arn arn:aws:iam::account:role/role-name-with-path
        """).build();

S3Client s3Client = S3Client.builder()
    .credentialsProvider(processCredentialsProvider)
    .build();
```

모든 아티팩트를 수집한 후 빨간색으로 표시된 텍스트를 값으로 대체합니다.

AWS SDK for Java 2.x를 사용하여 코드에 자격 증명 제공

기본 자격 증명 체인이나 특정 또는 사용자 지정 공급자 또는 공급자 체인이 애플리케이션에 작동하지 않는 경우 코드에서 직접 임시 자격 증명을 제공할 수 있습니다. 위에 설명된 IAM 역할 자격 증명 또는 AWS Security Token Service ()에서 검색된 임시 자격 증명일 수 있습니다AWS STS. 를 사용하여 임시 자격 증명을 검색한 경우 다음 코드 예제와 같이 AWS 서비스 클라이언트에 AWS STS제공합니다.

1. `StsClient.assumeRole()`를 호출하여 역할을 맡으세요.
2. StaticCredentialsProvider 객체를 생성하고 `AwsSessionCredentials` 객체와 함께 제공하세요.
3. `StaticCredentialsProvider`를 사용하여 클라이언트 빌더를 구성하고 클라이언트를 빌드합니다.

다음 예시에서는 IAM 수임 역할에 AWS STS 대해에서 반환한 임시 자격 증명을 사용하여 Amazon S3 서비스 클라이언트를 생성합니다.

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

권한 세트

에 정의된 다음 권한 세트를 AWS IAM Identity Center 사용하면 자격 증명(사용자)이 다음 두 작업을 수행할 수 있습니다.

1. Amazon Simple Storage Service `GetObject` 운영.
2. AWS Security Token Service의 `AssumeRole` 작업.

역할을 맡지 않으면 예제에 표시된 `s3.listBuckets()` 메서드가 실패합니다.

JSON

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

수임된 역할

수임된 역할 권한 정책

다음 권한 정책은 이전 예제에 수임된 역할에 연결되어 있습니다. 이 권한 정책은 역할과 동일한 계정의 모든 버킷을 나열하는 기능을 허용합니다.

JSON

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

```
}
```

수임된 역할 신뢰 정책

다음 권한 정책은 이전 예제에 수임된 역할에 연결되어 있습니다. 이 정책을 통해 두 계정의 ID(사용자)가 역할을 수임할 수 있습니다.

JSON

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

SDK for Java 2.x를 사용하여 Amazon EC2에서 IAM 역할 자격 증명 읽기

IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 AWS CLI 또는 AWS API 요청을 수행하는 애플리케이션의 임시 자격 증명을 관리할 수 있습니다. 이것은 EC2 인스턴스 내에 액세스 키를 저장하는 경우에 바람직한 방법입니다. EC2 인스턴스에 AWS 역할을 할당하고 모든 애플리케이션에서 사용할 수 있도록 하려면 인스턴스에 연결된 인스턴스 프로파일을 생성합니다. 인스턴스 프로필에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 보안 인증을 얻을 수 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하세요.

이 주제에서는 EC2 인스턴스에서 실행되도록 Java 애플리케이션을 설정하고가 IAM 역할 자격 증명을 획득 AWS SDK for Java 2.x 하도록 설정하는 방법에 대한 정보를 제공합니다.

환경에서 IAM 역할 자격 증명 획득

애플리케이션이 `create` 메서드(또는 `builder().build()` 메서드)를 사용하여 AWS 서비스 클라이언트를 생성하는 경우 Java용 SDK는 기본 자격 증명 공급자 체인을 사용합니다. 기본 자격 증명 공급자 체인은 SDK가 임시 자격 증명과 교환할 수 있는 구성 요소를 위한 실행 환경을 검색합니다. 이 the section called “기본 자격 증명 공급자 체인 사용” 섹션에서는 전체 검색 프로세스를 설명합니다.

기본 공급자 체인의 마지막 단계는 애플리케이션이 Amazon EC2 인스턴스에서 실행되는 경우에만 사용할 수 있습니다. 이 단계에서는 SDK가 `InstanceProfileCredentialsProvider`를 사용하여 EC2 인스턴스 프로필에 정의된 IAM 역할을 읽습니다. 그런 다음 SDK는 해당 IAM 역할에 대한 임시 보안 인증을 획득합니다.

이러한 자격 증명은 임시용이므로 기간이 경과되면 만료되지만, `InstanceProfileCredentialsProvider`는 가져온 자격 증명을 통해 계속 AWS에 액세스할 수 있도록 자격 증명을 정기적으로 새로 고칩니다.

프로그래밍 방식으로 IAM 역할 자격 증명 획득

최종적으로 EC2에서 `InstanceProfileCredentialsProvider`를 사용하는 기본 자격 증명 공급자 체인의 대안으로 `InstanceProfileCredentialsProvider`를 사용하여 서비스 클라이언트를 명시적으로 구성할 수 있습니다. 이 접근 방법은 다음 코드 조각에 나와 있습니다.

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

IAM 역할 자격 증명을 안전하게 획득

기본적으로 EC2 인스턴스는 구성된 IAM 역할과 같은 정보에 SDK의 `InstanceProfileCredentialsProvider`가 액세스할 수 있도록 IMDS(인스턴스 메타데이터 서비스)를 실행합니다. EC2 인스턴스는 기본적으로 2가지 버전의 IMDS를 실행합니다.

- 인스턴스 메타데이터 서비스 버전 1(IMDSv1) – 요청/응답 방법
- 인스턴스 메타데이터 서비스 버전 2(IMDSv2) – 세션 지향 방법

IMDSv2는 IMDSv1보다 더 안전한 접근 방식입니다.

기본적으로 Java SDK는 먼저 IMDSv2를 시도하여 IAM 역할을 가져오지만, 실패하면 IMDSv1을 시도합니다. 그러나 IMDSv1은 덜 안전하므로는 IMDSv2만 사용하고 SDK가 IMDSv1을 시도하지 못하도록 비활성화하는 것이 AWS 좋습니다.

보다 안전한 접근 방식을 사용하려면 다음 설정 중 하나에 `true` 값을 제공하여 SDK가 IMDSv1을 사용하지 않도록 설정합니다.

- 환경 변수: `AWS_EC2_METADATA_V1_DISABLED`
- JVM 시스템 속성: `aws.disableEc2MetadataV1`
- 공유 구성 파일 설정: `ec2_metadata_v1_disabled`

이러한 설정 중 하나를 `true`로 설정하면 초기 IMDSv2 직접 호출이 실패할 경우 SDK는 IMDSv1을 사용하여 IMDS 역할 자격 증명을 로드하지 않습니다.

AWS SDK for Java 2.x에서 재시도 동작 구성

AWS 서비스에 대한 호출은 예기치 않은 이유로 가끔 실패할 수 있습니다. 호출을 재시도하면 스로틀링(속도 초과) 또는 일시적 오류와 같은 특정 오류가 성공할 수 있습니다. AWS SDK for Java 2.x에는 이러한 오류를 감지하고 모든 클라이언트에 대해 기본적으로 사용 설정된 호출을 자동으로 재시도하는 메커니즘이 내장되어 있습니다.

이 페이지에서는 작동 방식, 고유한 모드를 구성하는 방법, 재시도 동작을 사용자 지정하는 방법을 설명합니다.

재시도 전략

재시도 전략은 SDK에서 재시도를 구현하는 데 사용되는 메커니즘입니다. 각 SDK 클라이언트에는 구축 시 만든 재시도 전략이 있으며, 클라이언트가 구축된 후에는 수정할 수 없습니다.

재시도 전략의 책임은 다음과 같습니다.

- 예외를 재시도 가능 여부로 분류합니다.
- 다음 시도 전에 대기하도록 제안된 지연을 계산합니다.
- 많은 비율의 요청이 실패하고 재시도가 실패할 때 재시도를 중지하는 메커니즘을 제공하는 토큰 버킷을 유지 관리합니다.

> **Note**
>
> SDK 버전 2.26.0으로 재시도 전략을 릴리스하기 전에 재시도 정책은 SDK에 재시도 메커니즘을 제공했습니다. 재시도 정책 API는 `software.amazon.awssdk.core.retry` 패키지의

주요 RetryPolicy 클래스로 구성된 반면, software.amazon.awssdk.retries 패키지에는 재시도 전략 API 요소가 포함되어 있습니다.
재시도 전략 API는 SDK의 주요 구성 요소의 인터페이스와 동작을 통합하기 위한 AWS의 포괄적 노력의 일환으로 도입되었습니다.

SDK for Java 2.x에는 표준, 레거시 및 적응형이라는 3가지 재시도 전략이 내장되어 있습니다. 3가지 재시도 전략 모두 재시도 가능한 예외 집합에 대해 재시도하도록 미리 구성되어 있습니다. 재시도 가능한 오류의 예로는 소켓 제한 시간, 서비스 측 스로틀링, 동시성 또는 낙관적 잠금 실패, 일시적인 서비스 오류가 있습니다.

표준 재시도 전략

표준 재시도 전략은 일반적인 사용 사례에 권장되는 RetryStrategy 구현입니다.
AdaptiveRetryStrategy와 달리 표준 전략은 일반적으로 모든 재시도 사용 사례에서 유용합니다.

기본적으로 표준 재시도 전략은 다음을 수행합니다.

- 구축 시 구성된 조건을 재시도합니다.
  StandardRetryStrategy.Builder#retryOnException를 사용하여 이 값을 조정할 수 있습니다.
- 총 3회 시도에서 2회 재시도합니다.
  StandardRetryStrategy.Builder#maxAttempts(int)를 사용하여 이 값을 조정할 수 있습니다.
- 스로틀링이 없는 예외의 경우 기본 지연 시간이 100밀리초이고 최대 지연 시간이 20초인 BackoffStrategy#exponentialDelay 백오프 전략을 사용합니다.
  StandardRetryStrategy.Builder#backoffStrategy를 사용하여 이 값을 조정할 수 있습니다.
- 스로틀링 예외의 경우 기본 지연 시간이 1초이고 최대 지연 시간이 20초인 BackoffStrategy#exponentialDelay 백오프 전략을 사용합니다.
  StandardRetryStrategy.Builder#throttlingBackoffStrategy를 사용하여 이 값을 조정할 수 있습니다.
- 다운스트림 장애가 높은 경우 회로 중단(재시도 사용 해제)을 수행합니다. 첫 번째 시도는 항상 실행되며 재시도만 사용 해제됩니다.
  StandardRetryStrategy.Builder#circuitBreakerEnabled로 조정합니다.

레거시 재시도 전략

레거시 재시도 전략은 일반적인 사용 사례용 RetryStrategy이지만, StandardRetryStrategy를 위해 더 이상 사용되지 않습니다. 이는 다른 전략을 지정하지 않을 때 클라이언트가 사용하는 기본 재시도 전략입니다.

스로틀링 예외와 비스로틀링 예외를 다르게 처리하는 것이 특징이며, 스로틀링 예외의 경우 백오프의 기본 지연은 비스로틀링 예외의 기본 지연(100ms)보다 크고(500ms) 스로틀링 예외는 토큰 버킷 상태에 영향을 미치지 않습니다.

내부에서 이 전략을 규모에 맞게 사용한 경험에 따르면 AWS가 표준 재시도 전략보다 더 나은 것은 아닙니다. 또한 다운스트림 서비스를 재시도 급증에서 보호하지 못하여 클라이언트 측에서 리소스 부족이 발생할 수 있습니다.

기본적으로 레거시 재시도 전략은 다음을 수행합니다.

- 구축 시 구성된 조건을 재시도합니다. LegacyRetryStrategy.Builder#retryOnException를 사용하여 이 값을 조정할 수 있습니다.
- 총 4회 시도에서 3회 재시도합니다. LegacyRetryStrategy.Builder#maxAttempts(int)를 사용하여 이 값을 조정할 수 있습니다.
- 스로틀링이 없는 예외의 경우 기본 지연 시간이 100밀리초이고 최대 지연 시간이 20초인 BackoffStrategy#exponentialDelay 백오프 전략을 사용합니다. LegacyRetryStrategy.Builder#backoffStrategy.를 사용하여 이 값을 조정할 수 있습니다.
- 스로틀링 예외의 경우 기본 지연 시간이 500밀리초이고 최대 지연 시간이 20초인 BackoffStrategy#exponentialDelay 백오프 전략을 사용합니다. LegacyRetryStrategy.Builder#throttlingBackoffStrategy를 사용하여 이 값을 조정할 수 있습니다.
- 다운스트림 장애가 높은 경우 회로 중단(재시도 사용 해제)을 수행합니다. 회로 차단으로 인해 첫 번째 시도가 성공할 수 없습니다. LegacyRetryStrategy.Builder#circuitBreakerEnabled를 사용하여 이 동작을 조정할 수 있습니다.
- 회로 차단의 상태는 스로틀링 예외의 영향을 받지 않습니다.

적응형 재시도 전략

적응형 재시도 전략은 리소스 제약이 많은 사용 사례의 `RetryStrategy`입니다.

적응형 재시도 전략에는 표준 전략의 모든 기능이 포함되며 비스로틀링 요청과 비교하여 스로틀링 요청의 속도를 측정하는 클라이언트 측 속도 제한 도구가 추가됩니다. 전략은 이 측정을 사용하여 안전한 대역폭 내에 유지하려는 시도로 요청을 늦춰 스로틀링 오류가 발생하지 않는 것이 이상적입니다.

기본적으로 적응형 재시도 전략은 다음을 수행합니다.

- 구축 시 구성된 조건을 재시도합니다. `AdaptiveRetryStrategy.Builder#retryOnException`를 사용하여 이 값을 조정할 수 있습니다.
- 총 3회 시도에서 2회 재시도합니다. `AdaptiveRetryStrategy.Builder#maxAttempts(int)`를 사용하여 이 값을 조정할 수 있습니다.
- 다운스트림 리소스에 대한 현재 로드를 기반으로 하는 동적 백오프 지연을 사용합니다.
- 다운스트림 실패 횟수가 많을 때 회로 차단(재시도 사용 해제)을 수행합니다. 회로 차단은 다운스트림 서비스를 보호하기 위해 중단 시나리오에서 두 번째 시도를 방지할 수 있습니다.

> **Warning**
>
> 적응형 재시도 전략은 클라이언트가 단일 리소스(예: DynamoDB 테이블 하나 또는 Amazon S3 버킷 하나)에 대해 작동한다고 가정합니다.
> 단일 클라이언트를 여러 리소스에서 사용하는 경우 한 리소스와 연결된 스로틀링 또는 중단으로 인해 클라이언트가 다른 모든 리소스에 액세스할 때 지연 시간이 증가하고 장애가 발생합니다. 적응형 재시도 전략을 사용하는 경우 각 리소스에 대해 단일 클라이언트를 사용하는 것이 좋습니다.
> 또한 모든 클라이언트가 리소스에 대해 적응형 재시도 전략을 사용하는 경우에도 이 전략을 사용하는 것이 좋습니다.

> **Important**
>
> Java SDK 2.26.0을 사용한 재시도 전략 릴리스에는 새로운 `RetryMode.ADAPTIVE_V2` 열거 값이 포함되어 있습니다. `ADAPTIVE_V2` 모드는 이전에 스로틀링 오류가 감지되었을 때 첫 번째 시도를 지연시키지 못한 오류를 수정합니다.

2.26.0 릴리스에서는 환경 변수, 시스템 속성 또는 프로필 설정에서 모드를 `adaptive`로 설정하여 사용자가 자동으로 `ADAPTIVE_V2` 모드 동작을 가져옵니다. 이러한 설정에는 `adaptive_v2` 값이 없습니다. 모드를 설정하는 방법은 다음 the section called “전략 지정” 섹션을 참조하세요.
사용자는 `RetryMode.ADAPTIVE`를 사용하여 코드에서 모드를 설정해 이전 동작을 가져올 수 있습니다.

요약: 재시도 전략 기본값 비교

다음 표에는 각 재시도 전략의 속성에 대한 기본값이 나와 있습니다.

Strategy	최대 시도 횟수	비스로틀링 오류의 기본 지연	스로틀링 오류의 기본 지연	토큰 버킷 크기	비스로틀링 재시도당 토큰 비용	스로틀링 재시도당 토큰 비용
표준	3	100ms	1000ms	500	5	5
레거시	4	100ms	500ms	500	5	0
적응형	3	100ms	100ms	500	5	5

Note

DynamoDB 클라이언트는 모든 재시도 전략에 대해 기본 최대 재시도 횟수(8회)를 사용하며, 이는 위의 표에 표시된 다른 AWS 서비스 클라이언트의 값보다 많습니다.

전략 지정

4가지 방법을 통해 서비스 클라이언트에 대한 전략을 지정할 수 있습니다.

코드에서

클라이언트를 구축할 때 재시도 전략을 사용하여 Lambda 표현식을 구성할 수 있습니다. 다음 코드 조각은 DynamoDB 서비스 클라이언트에서 기본값을 사용하는 표준 재시도 전략을 구성합니다.

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

```
        .overrideConfiguration(o -> o.retryStrategy(RetryMode.STANDARD))
        .build();
```

RetryMode.STANDARD 대신 RetryMode.LEGACY 또는 RetryMode.ADAPTIVE를 지정할 수 있습니다.

프로필 설정 기준

공유 AWS 구성 파일에 프로필 설정으로 retry_mode를 포함시킵니다. standard, legacy 또는 adaptive를 값으로 지정합니다. 프로필 설정으로 설정하면 프로필이 활성 상태인 동안 만들어진 모든 서비스 클라이언트는 기본값으로 지정된 재시도 전략을 사용합니다. 이전과 같이 코드에서 재시도 전략을 구성하여 이 설정을 재정의할 수 있습니다.

다음 프로필을 사용하면 모든 서비스 클라이언트가 표준 재시도 전략을 사용합니다.

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

JVM 시스템 속성 기준

코드에서 재정의되지 않는 한 시스템 속성 aws.retryMode를 사용하여 모든 서비스 클라이언트에 대해 재시도 상태를 구성할 수 있습니다. standard, legacy 또는 adaptive를 값으로 지정합니다.

다음 명령과 같이 Java를 호출할 때 -D 스위치를 사용합니다.

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

또는 다음 코드 조각과 같이 클라이언트를 만들기 전에 코드에서 시스템 속성을 설정합니다.

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

환경 변수 사용

또는 값이 standard, legacy 또는 adaptive인 AWS_RETRY_MODE 환경 변수를 사용할 수 있습니다. 프로필 설정 또는 JVM 시스템 속성과 마찬가지로 코드에서 클라이언트를 구성하지 않는 한 환경 변수는 지정된 재시도 모드로 모든 서비스 클라이언트를 구성합니다.

다음 명령은 현재 쉘 세션의 재시도 모드를 standard로 설정합니다.

```
export AWS_RETRY_MODE=standard
```

전략 사용자 지정

재시도 가능한 최대 시도 횟수, 백오프 전략 및 예외를 설정하여 모든 재시도 전략을 사용자 지정할 수 있습니다. 재시도 전략을 구축할 때 또는 구성된 전략을 추가로 구체화할 수 있는 재정의 빌더를 사용하여 클라이언트를 구축할 때 사용자 지정할 수 있습니다.

최대 시도 횟수 사용자 지정

다음 문과 같이 클라이언트 구문 중에 최대 시도 횟수를 구성할 수 있습니다. 다음 문은 클라이언트의 기본 재시도 전략을 최대 5회까지 사용자 지정합니다. 첫 번째 시도와 4회 재시도입니다.

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

또는 다음 코드 예제와 같이 전략을 구축하여 클라이언트에 제공할 수 있습니다. 다음 코드는 표준 최대 3회 시도를 10회로 대체하고 DynamoDB 클라이언트를 사용자 지정 전략으로 구성합니다.

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

> **Warning**
>
> 각 클라이언트를 고유한 RetryStrategy 인스턴스로 구성하는 것이 좋습니다. RetryStrategy 인스턴스를 공유하는 경우 한 클라이언트의 장애가 다른 클라이언트의 재시도 동작에 영향을 미칠 수 있습니다.

코드 대신 외부 설정을 사용하여 모든 클라이언트에 대한 최대 시도 횟수를 설정할 수도 있습니다. the section called “전략 지정” 섹션에 설명된 대로 이 설정을 구성합니다.

재시도 가능한 예외 사용자 지정

클라이언트 구문 중에 재시도를 트리거하는 추가 예외를 구성할 수 있습니다. 이 사용자 지정은 기본 재시도 가능 예외 세트에 포함되지 않은 예외가 발생하는 엣지 사례에 제공됩니다.

`retryOnException` 및 `retryOnExceptionOrCause` 메서드는 기존 재시도 가능한 예외 세트에 새 예외 유형을 추가하지만 기본 세트를 대체하지 않습니다. 이렇게 하면 SDK의 기본 재시도 기능을 유지하면서 재시도 동작을 확장할 수 있습니다.

SDK에서 직접 예외가 발생하거나 예외가 다른 예외의 원인으로 래핑되는 경우 `retryOnExceptionOrCause` 메서드는 재시도 가능한 예외를 추가합니다.

다음 코드 조각은 재시도 예외(`retryOnException` 및 `retryOnExceptionOrCause`)를 사용자 지정하는 데 사용하는 메서드를 보여줍니다. SDK에서 직접 예외가 발생하거나 예외가 래핑된 경우 `retryOnExceptionOrCause` 메서드는 재시도 가능한 예외를 추가합니다.

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

> **Important**
>
> 재시도 전략 구성에 관계없이 `subscribeToShard` 메서드를 호출하면 Kinesis 비동기식 클라이언트에 대해 재시도가 사용 해제됩니다.

백오프 전략 사용자 지정

백오프 전략을 구축하여 클라이언트에 제공할 수 있습니다.

다음 코드는 기본 표준 전략의 지수 지연 백오프 전략을 대체하는 `BackoffStrategy`를 구축합니다.

```
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();
```

`RetryPolicy`에서 `RetryStrategy`로 마이그레이션

`RetryPolicy`(재시도 정책 API)는 가까운 미래에 지원될 예정입니다. 현재 `RetryPolicy`의 인스턴스를 사용하여 클라이언트를 구성하는 경우 모든 사항이 이전과 같이 작동합니다. 백그라운드에서 Java SDK는 이를 `RetryStrategy`에 적용합니다. 새로운 재시도 전략 인터페이스는 `RetryPolicy`와 동일한 기능을 제공하지만 다르게 만들어지고 구성됩니다.

AWS SDK for Java 2.x에서 제한 시간 구성

AWS SDK for Java 2.x는 복원력이 뛰어난 애플리케이션을 구축하는 데 도움이 되는 여러 계층의 제한 시간 구성을 제공합니다. SDK는 애플리케이션의 성능과 신뢰성을 최적화하기 위해 함께 작동하는 다양한 유형의 제한 시간을 제공합니다.

SDK에는 2가지 주요 제한 시간 범주가 있습니다.

- 서비스 클라이언트 제한 시간 - API 작업을 제어하는 상위 수준 제한 시간
- HTTP 클라이언트 제한 시간 - 네트워크 통신을 제어하는 하위 수준 제한 시간

서비스 클라이언트 제한 시간

서비스 클라이언트 제한 시간은 API 수준에서 작동하며 재시도 및 여러 번의 시도를 포함하여 서비스 작업의 전반적인 동작을 제어합니다.

API 호출 제한 시간

API 호출 제한 시간은 모든 재시도를 포함하여 전체 API 작업의 최대 시간을 설정합니다. 이 제한 시간은 애플리케이션이 전체 작업을 완료할 때까지 기다리는 시간을 엄격하게 제한합니다.

```
S3Client s3Client = S3Client.builder()
    .overrideConfiguration(ClientOverrideConfiguration.builder()
        .apiCallTimeout(Duration.ofMinutes(2))  // Total time for entire operation,
 such as when you call the getObject method.
        .build())
    .build();
```

주요 특징:

- 모든 재시도를 포함합니다.
- 재시도 사이에 대기하는 데 소요된 시간을 포함합니다.
- 절대 최대 대기 시간을 제공합니다.
- 작업이 무기한으로 실행되지 않도록 합니다.

API 호출 시도 제한 시간

API 호출 시도 제한 시간은 API 작업의 단일 시도에 대한 최대 시간을 설정합니다. 이 제한 시간을 초과하면 SDK는 전체 호출에 실패하지 않고 작업을 재시도합니다(재시도가 구성된 경우).

```
S3Client s3Client = S3Client.builder()
    .overrideConfiguration(ClientOverrideConfiguration.builder()
        .apiCallAttemptTimeout(Duration.ofSeconds(30))  // Time for single attempt.
        .build())
    .build();
```

주요 특징:

- 각 시도에만 적용됩니다.
- 빠른 실패를 사용하고 느린 요청에 대해 다시 시도합니다.
- API 호출 제한 시간보다 짧아야 합니다.
- 일시적인 문제를 식별하고 복구하는 데 도움이 됩니다.

서비스 클라이언트 제한 시간 구성

모든 작업 또는 요청당 전역적으로 서비스 클라이언트 제한 시간을 구성할 수 있습니다.

전역 구성:

```
S3Client s3Client = S3Client.builder()
    .overrideConfiguration(b -> b
        .apiCallTimeout(Duration.ofSeconds(105L))
        .apiCallAttemptTimeout(Duration.ofSeconds(25L)))
    .build();
// When you use the s3Client for an API operation, the SDK uses the configured timeout
 values.
```

요청당 구성:

```
S3Client basicS3Client = S3Client.create();

// The following configuration uses the same settings as shown before, but these
 settings
// apply to only the `putObject` call. When you use `basicS3Client` in another API call
 without
// supplying the override configuration, there are no API timeout limits. No timeout
 limits is the default for the SDK.
AwsRequestOverrideConfiguration overrideConfiguration =
 AwsRequestOverrideConfiguration.builder()
    .apiCallTimeout(Duration.ofSeconds(105L))
    .apiCallAttemptTimeout(Duration.ofSeconds(25L))
    .build();

basicS3Client.putObject(b -> b
        .bucket("amzn-s3-demo-bucket")
        .key("example-key")
        .overrideConfiguration(overrideConfiguration),
    RequestBody.fromString("test"));
```

API 제한 시간에 대한 모범 사례

SDK for Java 2.x는 API 호출 제한 시간 또는 각 API 호출 시도 제한 시간에 기본값을 설정하지 않습니다. 각 시도와 전체 요청 모두에 대해 제한 시간을 설정하는 것이 좋습니다. 이렇게 하면 일시적인 문제로 인해 요청 시도 시간이 더 오래 걸리거나 치명적인 네트워크 문제가 발생할 때 애플리케이션이 빠르게 실패할 수 있습니다.

HTTP 클라이언트 제한 시간

HTTP 클라이언트 제한 시간은 네트워크 수준에서 작동하며 HTTP 통신의 다양한 측면을 제어합니다. 이러한 제한 시간은 사용하는 HTTP 클라이언트 구현에 따라 달라집니다.

연결 제한 시간

연결 제한 시간은 AWS 서비스 엔드포인트에 대한 새 연결을 설정할 때 대기하는 시간을 제어합니다.

```
// Available with all HTTP clients.
ApacheHttpClient.builder()
    .connectionTimeout(Duration.ofSeconds(5L))
```

```
    .build();
```

용도:

- 네트워크 연결 문제가 중단되는 것을 방지합니다.
- 서비스에 연결할 수 없을 때 빠르게 실패합니다.
- 응답 오류 처리가 필요한 애플리케이션에 필수입니다.

소켓 제한 시간(Apache 및 URLConnection 클라이언트)

소켓 제한 시간은 설정된 연결에서 데이터를 기다리는 시간을 제어합니다.

```
ApacheHttpClient.builder()
    .socketTimeout(Duration.ofSeconds(30L))  // Time to wait for response data.
    .build();
```

읽기 및 쓰기 제한 시간(Netty 클라이언트)

Netty 클라이언트는 읽기 및 쓰기 작업을 위한 별도의 제한 시간을 제공합니다.

```
NettyNioAsyncHttpClient.builder()
    .readTimeout(Duration.ofSeconds(30L))   // Reading response data.
    .writeTimeout(Duration.ofSeconds(30L))  // Writing request data.
    .build();
```

TLS 협상 제한 시간(Netty 클라이언트)

다음과 같이 TLS/SSL 핸드셰이크에 허용되는 시간을 제어합니다.

```
NettyNioAsyncHttpClient.builder()
    .tlsNegotiationTimeout(Duration.ofSeconds(3L))
    .build();
```

연결 풀 제한 시간

일부 HTTP 클라이언트는 연결 풀 작업에 제한 시간을 제공합니다.

```
ApacheHttpClient.builder()
```

```
    .connectionAcquisitionTimeout(Duration.ofSeconds(10L))  // Wait for pool
connection.
    .connectionTimeToLive(Duration.ofMinutes(5L))           // Maximum connection age.
    .connectionMaxIdleTime(Duration.ofSeconds(60L))         // Maximum idle time.
    .build()
```

HTTP 클라이언트 구성에는 AWS SDK for Java 2.x의 HTTP 클라이언트에 대한 자세한 정보가 포함되어 있습니다.

제한 시간 상호 작용 및 계층 구조

적절하게 구성하려면 서로 다른 제한 시간이 상호 작용하는 방식을 이해하는 것이 중요합니다.

제한 시간 계층 구조

```
API Call Timeout (2 minutes)
### Retry Attempt 1
#   ### API Call Attempt Timeout (45 seconds)
#   ### HTTP Client Timeouts
#       ### Connection Timeout (5 seconds)
#       ### TLS Negotiation Timeout (3 seconds)
#       ### Read/Write Timeout (30 seconds)
### Retry Attempt 2
#   ### [Same structure as Attempt 1]
### Retry Attempt 3
    ### [Same structure as Attempt 1]
```

구성 규칙

API 호출 제한 시간 ≥ API 호출 시도 제한 시간

```
// Correct configuration.
.apiCallTimeout(Duration.ofMinutes(2))         // 120 seconds.
.apiCallAttemptTimeout(Duration.ofSeconds(30)) // 30 seconds.
```

API 호출 시도 제한 시간 ≥ HTTP 클라이언트 제한 시간

```
// HTTP client timeouts must be less than attempt timeout.
.apiCallAttemptTimeout(Duration.ofSeconds(30L))   // 30 seconds.
// HTTP client configuration.
.connectionTimeout(Duration.ofSeconds(5L))        // 5 seconds.
```

```
.readTimeout(Duration.ofSeconds(25L))            // 25 seconds (< 30).
```

여러 차례 시도용 계정

```
// If you have 3 retry attempts, each taking up to 30 seconds
// API call timeout must be at least 90 seconds plus overhead.
.apiCallTimeout(Duration.ofMinutes(2L))          // 120 seconds.
.apiCallAttemptTimeout(Duration.ofSeconds(30))   // 30 seconds per attempt.
```

스마트 구성 기본값 사용

SDK는 적절한 제한 시간 값을 자동으로 구성하는 스마트 기본값을 제공합니다.

```
// Enable smart defaults.
S3Client client = S3Client.builder()
    .defaultsMode(DefaultsMode.AUTO)  // Automatically choose appropriate defaults.
    .build();

// Available modes:
// - STANDARD: Balanced defaults
// - IN_REGION: Optimized for same-region calls
// - CROSS_REGION: Optimized for cross-region calls
// - MOBILE: Optimized for mobile applications
// - AUTO: Automatically detect and choose appropriate mode
// - LEGACY: Provides settings that were used before smart defaults existed.
```

스마트 기본값은 다음을 자동으로 구성합니다.

- 연결 초과 시간 값입니다.
- TLS 협상 제한 시간 값입니다.
- 기타 클라이언트 설정입니다.

요약

AWS SDK for Java 2.x의 효과적인 제한 시간 구성을 사용하려면 서비스 클라이언트 제한 시간과 HTTP 클라이언트 제한 시간 간의 상호 작용을 이해해야 합니다.

1. 서비스 클라이언트 제한 시간은 상위 수준 API 동작을 제어합니다.

2. HTTP 클라이언트 제한 시간은 하위 수준 네트워크 동작을 제어합니다.

3. 적절하게 계층 구조를 구성하면 제한 시간이 효과적으로 함께 작동합니다.

4. 스마트 기본값은 대부분의 애플리케이션에 적합한 시작점을 제공합니다.

사용 사례에 맞게 제한 시간을 적절하게 구성하여 네트워크 문제에 복원력이 뛰어나고 사용자에게 응답하는 애플리케이션을 구축합니다.

AWS SDK for Java 2.x에서 관찰성 기능 구성

원격 측정은 시스템의 동작을 모니터링하고 분석할 수 있도록 원격 소스에서 데이터를 자동으로 수집하고 전송하는 기능입니다.

SDK for Java 2.x의 관찰성은 개발자에게 전체 요청 수명 주기를 캡처하는 풍부한 원격 측정 데이터를 통해 Java 애플리케이션이 AWS 서비스와 상호 작용하는 방식에 대한 포괄적인 인사이트를 제공합니다. 이러한 기능을 사용하면 지표, 로그 및 추적과 같은 원격 측정 신호를 수집, 분석 및 시각화하여 요청 패턴을 모니터링하고, 병목 현상을 식별하고, 애플리케이션의 AWS 상호 작용을 최적화하여 신뢰성과 성능을 개선할 수 있습니다.

주제

- AWS SDK for Java 2.x의 SDK 지표 게시
- AWS SDK for Java 2.x 애플리케이션 모니터링
- SDK를 사용하여 Java 2.x로 로깅하기

AWS SDK for Java 2.x의 SDK 지표 게시

AWS SDK for Java 2.x를 사용하면 애플리케이션의 서비스 클라이언트에 대한 지표를 수집하고 Amazon CloudWatch Logs의 출력을 분석한 다음 그에 따라 조치를 취할 수 있습니다.

기본적으로 SDK에서 지표 수집은 비활성화되어 있습니다. 이 항목은 활성화하고 구성하는 데 도움이 됩니다.

SDK 지표 사용 시작하기

애플리케이션에서 지표 수집을 사용하려면 사용 사례에 따라 `MetricPublisher` 인터페이스의 적절한 구현을 선택하고 자세한 설정 안내를 따릅니다.

장기 실행 애플리케이션의 경우:

- 사용CloudWatchMetricPublisher
- 전체 설정 안내, 코드 예제 및 구성 옵션은 장기 실행 애플리케이션의 SDK 지표 게시를 참조하세요.

AWS Lambda 함수의 경우:

- 사용EmfMetricLoggingPublisher
- 전체 설정 안내, 종속성 및 Lambda별 구성은 AWS Lambda 함수에 대한 SDK 지표 게시를 참조하세요.

문제 해결 및 콘솔 출력의 경우:

- 사용LoggingMetricPublisher
- 설정 안내, 형식 지정 옵션, 로컬 개발 및 문제 해결을 위한 예제는 개발 및 디버깅용 콘솔에 대한 SDK 지표 출력을 참조하세요.

빠른 구현 미리 보기

각 사용 사례에 대한 지표 사용 설정은 다음과 같습니다.

장기 실행 애플리케이션:

```
MetricPublisher metricsPub = CloudWatchMetricPublisher.create();
DynamoDbClient ddb = DynamoDbClient.builder()
    .overrideConfiguration(c -> c.addMetricPublisher(metricsPub))
    .build();
```

Lambda 함수

```
EmfMetricLoggingPublisher emfPublisher = EmfMetricLoggingPublisher.builder()
    .namespace("MyApp")
    .build();
DynamoDbClient dynamoDb = DynamoDbClient.builder()
    .overrideConfiguration(c -> c.addMetricPublisher(emfPublisher))
    .build();
```

개발 및 디버깅:

```
MetricPublisher loggingPublisher = LoggingMetricPublisher.create();
S3Client s3 = S3Client.builder()
    .overrideConfiguration(c -> c.addMetricPublisher(loggingPublisher))
    .build();
```

AWS CRT 기반 S3 클라이언트의 지표 제한 사항

AWS CRT 기반 S3 클라이언트는 현재 SDK 지표 수집을 지원하지 않습니다. AWS CRT 기반 S3 클라이언트 인스턴스의 빌더인 `S3CrtAsyncClientBuilder`는 지표 게시자를 구성하는 메서드를 제공하지 않습니다.

지표는 언제 사용할 수 있나요?

지표는 일반적으로 Java용 SDK가 지표를 생성한 후 5~10분 이내에 사용할 수 있습니다. 정확한 최신 지표를 확인하려면 Java 애플리케이션에서 지표를 전송한 후 최소 10분 후에 Cloudwatch를 확인하세요.

어떤 정보가 수집되나요?

지표 수집에는 다음이 포함됩니다.

- 성공 또는 실패 여부를 포함한 API 요청 수
- 반환된 예외를 포함하여 API 요청에서 호출한 AWS 서비스에 대한 정보
- 마샬링, 서명, HTTP 요청과 같은 다양한 작업에 소요되는 기간
- 열린 연결 수, 보류 중인 요청 수, 사용된 HTTP 클라이언트 이름과 같은 HTTP 클라이언트 측정항목

> **Note**
>
> 사용 가능한 지표는 HTTP 클라이언트마다 다릅니다.

전체 목록은 서비스 클라이언트 메트릭을 참조하세요.

이 정보를 어떻게 사용할 수 있나요?

SDK가 수집하는 측정항목을 사용하여 애플리케이션의 서비스 클라이언트를 모니터링할 수 있습니다. 전반적인 사용 추세를 살펴보고, 이상 현상을 식별하고, 반환된 서비스 클라이언트 예외를 검토하거나,

특정 문제를 이해하기 위해 자세히 알아볼 수 있습니다. Amazon CloudWatch Logs를 사용하면 애플리케이션이 정의한 조건에 도달하는 즉시 알림을 보내는 경보를 만들 수도 있습니다.

자세한 내용은 Amazon CloudWatch Logs 사용 설명서에서 Amazon CloudWatch 지표 사용 및 Amazon CloudWatch 경보 사용을 참조하세요.

AWS SDK for Java 2.x를 사용하여 장기 실행 애플리케이션의 SDK 지표 게시

`CloudWatchMetricPublisher` 구현은 지연과 지표를 집계하고 지연되어 Amazon CloudWatch에 주기적으로 업로드하기 때문에 장기 실행 애플리케이션에 가장 적합합니다.

지표 게시자의 기본 설정은 메모리 사용량과 CloudWatch 비용을 최소화하면서 지표 데이터에 대한 유용한 인사이트를 제공하기 위한 것입니다.

설정

`CloudWatchMetricPublisher`를 통해 지표를 사용 설정하고 사용하려면 먼저 다음 단계를 완료해야 합니다.

1단계: 필수 종속성 추가

AWS SDK for Java 버전 `2.14.0` 또는 그 이상의 버전을 사용하도록 프로젝트 종속성(예: `pom.xml` 또는 `build.gradle` 파일)을 구성하세요.

프로젝트 종속 항목에 버전 번호 `2.14.0` 이상의 `cloudwatch-metric-publisher` artifactId를 포함해야 합니다.

예:

```
<project>
  <dependencyManagement>
   <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>2.30.11</version>  <!-- Navigate the link to see the latest version.
 -->
        <type>pom</type>
        <scope>import</scope>
      </dependency>
   </dependencies>
  </dependencyManagement>
  <dependencies>
```

```
    <dependency>
       <groupId>software.amazon.awssdk</groupId>
       <artifactId>cloudwatch-metric-publisher</artifactId>
    </dependency>
  </dependencies>
</project>
```

2단계: 필수 권한 구성

SDK for Java에서 지표 게시자를 사용하여 IAM ID에 대한 `cloudwatch:PutMetricData` 권한을 사용합니다.

특정 요청에 대한 지표 활성화

다음 클래스는 Amazon DynamoDB에 대한 요청에서 CloudWatch 지표 게시자를 사용하는 방법을 보여줍니다. 기본 메트릭 게시자 구성을 사용합니다.

```
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**
>
> 서비스 클라이언트가 더 이상 사용되지 않을 때 애플리케이션이 `MetricPublisher` 인스턴스에서 `close`를 호출하는지 확인합니다. 이렇게 하지 않으면 스레드 또는 파일 설명자 누수가 발생할 수 있습니다.

특정 서비스 클라이언트에 대한 요약 지표 사용

다음 코드 조각은 서비스 클라이언트의 기본 설정으로 CloudWatch 지표 게시자를 사용하는 방법을 보여줍니다.

```
MetricPublisher metricsPub = CloudWatchMetricPublisher.create();

DynamoDbClient ddb = DynamoDbClient.builder()
          .overrideConfiguration(c -> c.addMetricPublisher(metricsPub))
          .build();
```

CloudWatch 지표 게시자 사용자 지정

다음 클래스는 특정 서비스 클라이언트의 지표 게시자에 대한 사용자 지정 구성을 사용하는 방법을 보여줍니다. 사용자 정의에는 특정 프로필 로드, 지표 게시자가 요청을 전송하는 AWS 리전 지정, 게시자가 CloudWatch에 지표를 보내는 빈도 사용자 정의가 포함됩니다.

```
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();
    }
}
```

이전 코드 조각에 표시된 사용자 지정은 다음과 같은 효과가 있습니다.

- `cloudWatchClient` 메서드를 사용하면 지표를 전송하는 데 사용되는 CloudWatch 클라이언트를 사용자 지정할 수 있습니다. 이 예제에서는 클라이언트가 지표를 보내는 us-east-1 기본값과 다른 리전을 사용합니다. 또한 CloudWatch에 대한 요청을 인증하는 데 사용되는 자격 증명인 cloudwatch라는 다른 이름의 프로필을 사용합니다. 이러한 자격 증명에는 `cloudwatch:PutMetricData` 작업에 대한 권한이 있어야 합니다.

- `uploadFrequency` 메서드를 사용하면 지표 게시자가 CloudWatch에 지표를 업로드하는 빈도를 지정할 수 있습니다. 기본값은 분당 1회입니다.
- `maximumCallsPerUpload` 메서드는 업로드당 발생하는 호출 수를 제한합니다. 기본값은 무제한입니다.
- 기본적으로 SDK for Java 2.x는 네임스페이스(`AwsSdk/JavaSdk2`) 아래에 지표를 게시합니다. `namespace` 메서드를 사용하여 다른 값을 지정할 수 있습니다.
- 기본적으로 SDK는 요약 지표를 게시합니다. 요약 지표는 평균, 최소, 최대, 합계 및 샘플 수로 구성됩니다. SDK는 `detailedMetrics` 메서드에 하나 이상의 SDK 지표를 지정하여 각 지표에 대한 추가 데이터를 게시합니다. 이 추가 데이터는 CloudWatch에서 쿼리할 수 있는 p90 및 p99와 같은 백분위수 통계를 사용합니다. 세부 지표는 SDK 클라이언트 요청의 종단간 지연 시간을 측정하는 `APICallDuration`과 같은 지연 시간 지표에 특히 유용합니다. `CoreMetric` 클래스의 필드를 사용하여 다른 일반적인 SDK 지표를 지정할 수 있습니다.

다음 단계: Lambda 함수로 작업하는 경우 EMF 기반 지표 게시를 위한 AWS Lambda 함수에 대한 SDK 지표 게시를 참조하세요.

AWS SDK for Java 2.x를 사용하여 AWS Lambda 함수에 대한 SDK 지표 게시

Lambda 함수는 일반적으로 밀리초에서 분 단위 동안 실행되므로 `CloudWatchMetricPublisher`에서 발생하는 지표 전송 지연으로 인해 데이터가 손실될 위험이 있습니다.

`EmfMetricLoggingPublisher`는 지표를 CloudWatch Embedded Metric Format(EMF)의 정형화된 로그 항목으로 즉시 작성하여 보다 적합한 접근 방식을 제공합니다. `EmfMetricLoggingPublisher`는 AWS Lambda 및 Amazon Elastic Container Service등 Amazon CloudWatch Logs와의 통합이 내장된 실행 환경에서 작동합니다.

설정

`EmfMetricLoggingPublisher`를 통해 지표를 사용 설정하고 사용하려면 먼저 다음 단계를 완료해야 합니다.

1단계: 필수 종속성 추가

AWS SDK for Java 버전 `2.30.3` 또는 그 이상의 버전을 사용하도록 프로젝트 종속성(예: `pom.xml` 또는 `build.gradle` 파일)을 구성하세요.

프로젝트 종속 항목에 버전 번호 `2.30.3` 이상의 `emf-metric-logging-publisher` artifactId를 포함해야 합니다.

예:

```
<project>
  <dependencyManagement>
   <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>2.30.11</version>  <!-- Navigate the link to see the latest version.
 -->
        <type>pom</type>
        <scope>import</scope>
      </dependency>
   </dependencies>
  </dependencyManagement>
  <dependencies>
   <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>emf-metric-logging-publisher</artifactId>
   </dependency>
  </dependencies>
</project>
```

2단계: 필수 권한 구성

SDK for Java에서 지표 게시자를 사용하여 IAM ID에 대한 `logs:PutLogEvents` 권한을 사용 설정해 EMF 형식 로그를 작성합니다.

3단계: 로깅 설정

적절한 지표 수집을 확보하려면 `INFO` 수준 이하(예: `DEBUG`)에서 콘솔에 출력하도록 로깅을 구성합니다. `log4j2.xml` 파일의 경우:

```
<Loggers>
  <Root level="WARN">
   <AppenderRef ref="ConsoleAppender"/>
  </Root>
  <Logger
 name="software.amazon.awssdk.metrics.publishers.emf.EmfMetricLoggingPublisher"
 level="INFO" />
</Loggers>
```

`log4j2.xml` 파일 설정 방법에 대한 자세한 내용은 이 가이드의 로깅 주제를 참조하세요.

`EmfMetricLoggingPublisher` 구성 및 사용

다음 Lambda 함수 클래스는 먼저 `EmfMetricLoggingPublisher` 인스턴스를 만들고 구성한 다음 Amazon DynamoDB 서비스 클라이언트와 함께 사용합니다.

```
public class GameIdHandler implements RequestHandler<Map<String, String>, String> {
    private final EmfMetricLoggingPublisher emfPublisher;
    private final DynamoDbClient dynamoDb;

    public GameIdHandler() {
        // Build the publisher.
        this.emfPublisher = EmfMetricLoggingPublisher.builder()
                .namespace("namespace")
                .dimensions(CoreMetric.SERVICE_ID,
                        CoreMetric.OPERATION_NAME)
                .build();
        // Add the publisher to the client.
        this.dynamoDb = DynamoDbClient.builder()
                .overrideConfiguration(c -> c.addMetricPublisher(emfPublisher))
                .region(Region.of(System.getenv("AWS_REGION")))
                .build();
    }

    @Override
    public String handleRequest(Map<String, String> event, Context context) {
        Map<String, AttributeValue> gameItem = new HashMap<>();

        gameItem.put("gameId", AttributeValue.builder().s(event.get("id")).build());

        PutItemRequest putItemRequest = PutItemRequest.builder()
                .tableName("games")
                .item(gameItem)
                .build();

        dynamoDb.putItem(putItemRequest);

        return "Request handled";
    }
}
```

DynamoDB 클라이언트는 `putItem` 메서드를 실행할 경우 지표를 EMF 형식의 CloudWatch 로그 스트림에 자동으로 게시합니다.

EMF 로그 이벤트의 예제

예를 들어 다음과 같이 구성된 로깅을 사용하여 GameHandler Lambda 함수에 다음 이벤트를 전송하는 경우입니다.

```
{
  "id": "23456"
}
```

함수에서 이벤트를 처리한 후 다음 예제와 유사한 2개의 로그 이벤트를 찾습니다. 두 번째 이벤트의 JSON 객체에는 DynamoDB에 대한 `PutItem` 작업의 Java SDK 지표 데이터가 포함됩니다.

CloudWatch는 EMF 형식의 로그 이벤트를 수신하면 정형화된 JSON을 자동으로 구문 분석하여 지표 데이터를 추출합니다. 그런 다음 CloudWatch는 원본 로그 항목을 CloudWatch Logs에 저장하는 동안 해당 지표를 만듭니다.

```
2025-07-11 15:58:30 [main] INFO  org.example.GameIdHandler:39 - Received map:
 {id=23456}

2025-07-11 15:58:34 [main] INFO
 software.amazon.awssdk.metrics.publishers.emf.EmfMetricLoggingPublisher:43 -
{
    "_aws": {
        "Timestamp": 1752249513975,
        "LogGroupName": "/aws/lambda/GameId",
        "CloudWatchMetrics": [
            {
                "Namespace": "namespace",
                "Dimensions": [
                    [
                        "OperationName",
                        "ServiceId"
                    ]
                ],
                "Metrics": [
                    {
                        "Name": "AvailableConcurrency"
                    },
                    {
                        "Name": "PendingConcurrencyAcquires"
                    },
                    {
                        "Name": "ServiceCallDuration",
```

```
            "Unit": "Milliseconds"
        },
        {
            "Name": "EndpointResolveDuration",
            "Unit": "Milliseconds"
        },
        {
            "Name": "MaxConcurrency"
        },
        {
            "Name": "BackoffDelayDuration",
            "Unit": "Milliseconds"
        },
        {
            "Name": "MarshallingDuration",
            "Unit": "Milliseconds"
        },
        {
            "Name": "LeasedConcurrency"
        },
        {
            "Name": "SigningDuration",
            "Unit": "Milliseconds"
        },
        {
            "Name": "ConcurrencyAcquireDuration",
            "Unit": "Milliseconds"
        },
        {
            "Name": "ApiCallSuccessful"
        },
        {
            "Name": "RetryCount"
        },
        {
            "Name": "UnmarshallingDuration",
            "Unit": "Milliseconds"
        },
        {
            "Name": "ApiCallDuration",
            "Unit": "Milliseconds"
        },
        {
            "Name": "CredentialsFetchDuration",
```

```
                        "Unit": "Milliseconds"
                    }
                ]
            }
        ]
    },
    "AvailableConcurrency": 0,
    "PendingConcurrencyAcquires": 0,
    "OperationName": "PutItem",
    "ServiceCallDuration": 1339,
    "EndpointResolveDuration": 81,
    "MaxConcurrency": 50,
    "BackoffDelayDuration": 0,
    "ServiceId": "DynamoDB",
    "MarshallingDuration": 181,
    "LeasedConcurrency": 1,
    "SigningDuration": 184,
    "ConcurrencyAcquireDuration": 83,
    "ApiCallSuccessful": 1,
    "RetryCount": 0,
    "UnmarshallingDuration": 85,
    "ApiCallDuration": 1880,
    "CredentialsFetchDuration": 138
}
```

`EmfMetricLoggingPublisher.Builder`의 API 설명서에는 사용할 수 있는 구성 옵션이 나와 있습니다.

또한 CloudWatchMetricPublisher에 표시된 대로 단일 요청에 대해 EMF 지표 로깅을 사용할 수 있습니다.

다음 단계: 장기 실행 애플리케이션의 경우 CloudWatch 기반 지표 게시를 위해 장기 실행 애플리케이션의 SDK 지표 게시를 참조하세요.

AWS SDK for Java 2.x를 사용하여 SDK 지표를 콘솔로 출력

`LoggingMetricPublisher` 구현은 지표를 애플리케이션의 콘솔 또는 로그 파일에 직접 출력합니다. 이 접근 방식은 Amazon CloudWatch와 같은 외부 서비스 없이 SDK가 수집하는 지표를 개발, 디버깅 및 이해하는 데 적합합니다.

`CloudWatchMetricPublisher` 및 `EmfMetricLoggingPublisher`와 달리 `LoggingMetricPublisher`는 지연이나 외부 종속성 없이 즉각적인 출력을 제공합니다. 따라서 로컬 개발 및 문제 해결 시나리오에 적합합니다.

LoggingMetricPublisher를 사용해야 하는 경우

다음 작업 시 `LoggingMetricPublisher`를 사용합니다.

- 개발 중 지표 수집 디버그
- SDK가 작업에 대해 수집하는 지표 이해
- 로컬 방식으로 성능 문제 해결
- 외부 서비스 종속성 없이 지표 수집 테스트
- 콘솔 또는 로그 파일에서 지표 즉시 보기

> **Note**
>
> `LoggingMetricPublisher`는 영구 지표 스토리지 및 분석 기능이 필요한 프로덕션 환경에는 권장되지 않습니다.

지표용 콘솔 로깅 설정

`LoggingMetricPublisher` 출력을 보려면 `INFO` 레벨 메시지를 표시하도록 로깅 프레임워크를 구성합니다. 다음 `log4j2.xml` 구성을 사용하면 지표가 콘솔에 표시됩니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="WARN">
    <Appenders>
        <Console name="ConsoleAppender" target="SYSTEM_OUT">
            <PatternLayout pattern="%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg
%n"/>
        </Console>
    </Appenders>
    <Loggers>
        <Root level="INFO">
            <AppenderRef ref="ConsoleAppender"/>
        </Root>
        <!-- Ensure LoggingMetricPublisher output appears. -->
        <Logger name="software.amazon.awssdk.metrics.LoggingMetricPublisher"
 level="INFO" />
    </Loggers>
</Configuration>
```

이 구성은 SDK가 INFO 레벨의 콘솔에 지표를 출력하도록 지시합니다. LoggingMetricPublisher 로거 구성을 사용하면 루트 로거가 WARN 또는 ERROR 등의 상위 수준을 사용하더라도 지표 출력이 나타납니다.

서비스 클라이언트에 대한 콘솔 지표 사용

다음 예제에서는 LoggingMetricPublisher를 만들고 Amazon Simple Storage Service 클라이언트와 함께 사용하는 방법을 보여줍니다.

```
import software.amazon.awssdk.metrics.LoggingMetricPublisher;
import software.amazon.awssdk.metrics.MetricPublisher;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;

// Create a LoggingMetricPublisher with default settings.
MetricPublisher metricPublisher = LoggingMetricPublisher.create();

// Add the publisher to your service client.
S3Client s3Client = S3Client.builder()
    .region(Region.US_EAST_1)
    .overrideConfiguration(config -> config.addMetricPublisher(metricPublisher))
    .build();

// Make requests - metrics will appear in your console.
s3Client.listBuckets();

// Clean up resources.
metricPublisher.close();
s3Client.close();
```

지표 출력 형식 선택

LoggingMetricPublisher에서 지원되는 2가지 출력 형식:

- PLAIN 형식(기본값): 지표를 압축된 한 줄 항목으로 출력합니다.
- PRETTY 형식: 사람이 읽을 수 있는 여러 줄 형식으로 지표를 출력합니다.

다음 예제에서는 개발 중에 읽기 쉽도록 PRETTY 형식을 사용하는 방법을 보여줍니다.

```
import org.slf4j.event.Level;
import software.amazon.awssdk.metrics.LoggingMetricPublisher;
```

```
// Create a LoggingMetricPublisher with PRETTY format.
MetricPublisher prettyMetricPublisher = LoggingMetricPublisher.create(
    Level.INFO,
    LoggingMetricPublisher.Format.PRETTY
);

// Use with your service client.
S3Client s3Client = S3Client.builder()
    .region(Region.US_EAST_1)
    .overrideConfiguration(config -> config.addMetricPublisher(prettyMetricPublisher))
    .build();
```

전체 예제

다음의 예제는 2가지 방식으로 LoggingMetricPublisher를 사용하는 쿼리를 보여줍니다.

- 서비스 클라이언트 수준인 경우
- 단일 요청인 경우

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.slf4j.event.Level;
import software.amazon.awssdk.metrics.LoggingMetricPublisher;
import software.amazon.awssdk.metrics.MetricPublisher;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;

/**
 * Demonstrates how to use LoggingMetricPublisher with AWS S3 SDK for Java 2.x.
 * <p>
 * This demo focuses on the S3 listBuckets operation to show how metrics are collected
 * and logged to the console for development and debugging purposes.
 * <p>
 * LoggingMetricPublisher is ideal for:
 * - Development and debugging
 * - Console output for troubleshooting
 * - Understanding what metrics are being collected
 * - Testing metric collection without external dependencies
 */
public class S3LoggingMetricPublisherDemo {
```

```
   private static final Logger logger =
LoggerFactory.getLogger(S3LoggingMetricPublisherDemo.class);

   public static void main(String[] args) {
       S3LoggingMetricPublisherDemo demo = new S3LoggingMetricPublisherDemo();
       demo.demonstrateUsage();
   }

   /**
    * Demonstrates basic usage with S3Client and metrics enabled at the client level.
    */
   private void demonstrateUsage() {

       // Create a LoggingMetricPublisher with default settings. The SDK logs metrics
as text in a single line.
       // The default settings are equivalent to using
`LoggingMetricPublisher.Format.PLAIN`.

       MetricPublisher metricPublisher = LoggingMetricPublisher.create();

       // Create an S3 client with metrics enabled.
       try (S3Client s3Client = S3Client.builder()
               .region(Region.US_EAST_1)
               .overrideConfiguration(config ->
config.addMetricPublisher(metricPublisher))
               .build()) {

           // Make the listBuckets request - metrics will be logged to console.
           ListBucketsResponse response =
s3Client.listBuckets(ListBucketsRequest.builder().build());

           // The next block shows the using a different LoggingMetricPublisher with a
`PRETTY` format.
           // Since the metric publisher is added to the request using the
`overrideConfiguration`, this formatting
           // applies only to the one request.
           try {
               s3Client.listBuckets(ListBucketsRequest.builder()
                       .overrideConfiguration(config -> config
                               .addMetricPublisher(LoggingMetricPublisher.create(
                                       Level.INFO,
LoggingMetricPublisher.Format.PRETTY)))
                       .build());
```

```
            } catch (Exception e) {
                logger.info("Request failed with metrics logged: {}", e.getMessage());
            }
            logger.info("Found {} buckets in your AWS account.",
 response.buckets().size());

        } catch (Exception e) {
            logger.error("Error during S3 operation: {}", e.getMessage());
            logger.info("Note: This is expected if AWS credentials are not
 configured.");
        }

        // Close the metric publisher to flush any remaining metrics.
        metricPublisher.close();
    }
}
```

코드는 콘솔에 다음을 로그합니다.

```
INFO  LoggingMetricPublisher - Metrics published: MetricCollection(name=ApiCall,
 metrics=[MetricRecord(metric=MarshallingDuration, value=PT0.005409792S),
 MetricRecord(metric=RetryCount, value=0), MetricRecord(metric=ApiCallSuccessful,
 value=true), MetricRecord(metric=OperationName, value=ListBuckets),
 MetricRecord(metric=EndpointResolveDuration, value=PT0.000068S),
 MetricRecord(metric=ApiCallDuration, value=PT0.163802958S),
 MetricRecord(metric=CredentialsFetchDuration, value=PT0.145686542S),
 MetricRecord(metric=ServiceEndpoint, value=https://s3.amazonaws.com),
 MetricRecord(metric=ServiceId, value=S3)],
 children=[MetricCollection(name=ApiCallAttempt,
 metrics=[MetricRecord(metric=TimeToFirstByte, value=PT0.138816S),
 MetricRecord(metric=SigningDuration, value=PT0.007803459S),
 MetricRecord(metric=ReadThroughput, value=165153.96002660287),
 MetricRecord(metric=ServiceCallDuration, value=PT0.138816S),
 MetricRecord(metric=AwsExtendedRequestId, value=e13Swj3uwn0qP1Oz
+m7II5OGq7jf8xxT8H18iDfRBCQmDg+gU4ek91Xrsl8XxRLROlIzCAPQtsQF0DAAWOb8ntuKCzX2AJdj),
 MetricRecord(metric=HttpStatusCode, value=200),
 MetricRecord(metric=BackoffDelayDuration, value=PT0S),
 MetricRecord(metric=TimeToLastByte, value=PT0.148915667S),
 MetricRecord(metric=AwsRequestId, value=78AW9BM7SWR6YMGB)],
 children=[MetricCollection(name=HttpClient,
 metrics=[MetricRecord(metric=MaxConcurrency, value=50),
 MetricRecord(metric=AvailableConcurrency, value=0),
 MetricRecord(metric=LeasedConcurrency, value=1),
```

```
 MetricRecord(metric=ConcurrencyAcquireDuration, value=PT0.002623S),
 MetricRecord(metric=PendingConcurrencyAcquires, value=0),
 MetricRecord(metric=HttpClientName, value=Apache)], children=[])])])
INFO  LoggingMetricPublisher - [4e6f2bb5] ApiCall
INFO  LoggingMetricPublisher - [4e6f2bb5] ###############################################
INFO  LoggingMetricPublisher - [4e6f2bb5] # MarshallingDuration=PT0.000063S          #
INFO  LoggingMetricPublisher - [4e6f2bb5] # RetryCount=0                            #
INFO  LoggingMetricPublisher - [4e6f2bb5] # ApiCallSuccessful=true                  #
INFO  LoggingMetricPublisher - [4e6f2bb5] # OperationName=ListBuckets               #
INFO  LoggingMetricPublisher - [4e6f2bb5] # EndpointResolveDuration=PT0.000024375S   #
INFO  LoggingMetricPublisher - [4e6f2bb5] # ApiCallDuration=PT0.018463083S          #
INFO  LoggingMetricPublisher - [4e6f2bb5] # CredentialsFetchDuration=PT0.000022334S  #
INFO  LoggingMetricPublisher - [4e6f2bb5] # ServiceEndpoint=https://s3.amazonaws.com #
INFO  LoggingMetricPublisher - [4e6f2bb5] # ServiceId=S3                            #
INFO  LoggingMetricPublisher - [4e6f2bb5] ###############################################
INFO  LoggingMetricPublisher - [4e6f2bb5]     ApiCallAttempt
INFO  LoggingMetricPublisher - [4e6f2bb5]
 ###########################################################################################
INFO  LoggingMetricPublisher - [4e6f2bb5]     # TimeToFirstByte=PT0.0165575S
                                                                          #
INFO  LoggingMetricPublisher - [4e6f2bb5]     # SigningDuration=PT0.000301125S
                                                                          #
INFO  LoggingMetricPublisher - [4e6f2bb5]     # ReadThroughput=1195591.792850103
                                                                          #
INFO  LoggingMetricPublisher - [4e6f2bb5]     # ServiceCallDuration=PT0.0165575S
                                                                          #
INFO  LoggingMetricPublisher - [4e6f2bb5]     #
 AwsExtendedRequestId=3QI1eenRuokdszWqZBmBMDUmko6FlSmHkM
+CUMNMeLor7gJml4D4lv6QXUZ1zWoTgG+tHbr6yo2vHdz4h1P8PDovvtMFRCeB #
INFO  LoggingMetricPublisher - [4e6f2bb5]     # HttpStatusCode=200
                                                                          #
INFO  LoggingMetricPublisher - [4e6f2bb5]     # BackoffDelayDuration=PT0S
                                                                          #
INFO  LoggingMetricPublisher - [4e6f2bb5]     # TimeToLastByte=PT0.017952625S
                                                                          #
INFO  LoggingMetricPublisher - [4e6f2bb5]     # AwsRequestId=78AVFAF795AAWAXH
                                                                          #
INFO  LoggingMetricPublisher - [4e6f2bb5]
 ###########################################################################################
INFO  LoggingMetricPublisher - [4e6f2bb5]         HttpClient
INFO  LoggingMetricPublisher - [4e6f2bb5]
 ########################################
INFO  LoggingMetricPublisher - [4e6f2bb5]         # MaxConcurrency=50
   #
```

```
INFO  LoggingMetricPublisher - [4e6f2bb5]         # AvailableConcurrency=0
   #
INFO  LoggingMetricPublisher - [4e6f2bb5]         # LeasedConcurrency=1
   #
INFO  LoggingMetricPublisher - [4e6f2bb5]         #
 ConcurrencyAcquireDuration=PT0.00004S #
INFO  LoggingMetricPublisher - [4e6f2bb5]         # PendingConcurrencyAcquires=0
   #
INFO  LoggingMetricPublisher - [4e6f2bb5]         # HttpClientName=Apache
   #
INFO  LoggingMetricPublisher - [4e6f2bb5]
 ########################################
INFO  S3LoggingMetricPublisherDemo - Found 6 buckets in your AWS account.
```

예제용 추가 아티팩트

Maven pom.xml 파일

```
<?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>com.example</groupId>
    <artifactId>s3-logging-metric-publisher-demo</artifactId>
    <version>1.0.0</version>
    <packaging>jar</packaging>

    <name>AWS S3 LoggingMetricPublisher Demo</name>
    <description>Demonstrates how to use LoggingMetricPublisher with AWS S3 SDK for
 Java 2.x</description>

    <properties>
        <maven.compiler.source>17</maven.compiler.source>
        <maven.compiler.target>17</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <aws.java.sdk.version>2.31.66</aws.java.sdk.version>
        <log4j.version>2.24.3</log4j.version>
    </properties>

    <dependencyManagement>
        <dependencies>
```

```
            <!-- AWS SDK BOM for dependency management -->
            <dependency>
                <groupId>software.amazon.awssdk</groupId>
                <artifactId>bom</artifactId>
                <version>${aws.java.sdk.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>

            <!-- Log4j BOM for logging dependency management -->
            <dependency>
                <groupId>org.apache.logging.log4j</groupId>
                <artifactId>log4j-bom</artifactId>
                <version>${log4j.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <dependencies>
        <!-- AWS S3 SDK for demonstration -->
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>s3</artifactId>
        </dependency>

        <!-- Log4j2 SLF4J implementation -->
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-slf4j2-impl</artifactId>
        </dependency>

        <!-- Log4j2 Core -->
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-core</artifactId>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
```

```
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.13.0</version>
                <configuration>
                    <source>17</source>
                    <target>17</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

Log4j2.xml 구성 파일

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="WARN">
    <Appenders>
        <Console name="ConsoleAppender" target="SYSTEM_OUT">
            <PatternLayout pattern="%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg
%n"/>
        </Console>
    </Appenders>
    <Loggers>
        <Root level="INFO">
            <AppenderRef ref="ConsoleAppender"/>
        </Root>
        <!-- Ensure LoggingMetricPublisher output appears. -->
        <Logger name="software.amazon.awssdk.metrics.LoggingMetricPublisher"
 level="INFO"/>
    </Loggers>
</Configuration>
```

지표에는 애플리케이션의 AWS API 사용 패턴을 이해하는 데 도움이 되는 타이밍 정보, 서비스 세부 정보, 작업 이름 및 HTTP 상태 코드가 포함됩니다.

다음 단계

개발 및 디버깅에 LoggingMetricPublisher를 사용한 후 프로덕션 환경에 대해 다음 옵션을 고려합니다.

- 장기 실행 애플리케이션의 경우 CloudWatchMetricPublisher를 사용하여 분석 및 알림용으로 Amazon CloudWatch에 지표를 전송합니다.

- AWS Lambda 함수의 경우 EmfMetricLoggingPublisher를 사용하여 CloudWatch 임베디드 지표 형식으로 지표 게시

AWS SDK for Java 2.x: 포괄적인 지표 참조

AWS SDK for Java 2.x를 사용하면 애플리케이션의 서비스 클라이언트에서 지표를 수집한 다음 Amazon CloudWatch에 해당 지표를 게시(출력) 할 수 있습니다.

다음 표에는 수집할 수 있는 지표와 HTTP 클라이언트 사용 요구 사항이 나열되어 있습니다.

SDK 지표를 활성화 및 구성하는 자세한 내용은 SDK 지표 활성화를 참조하세요.

각 요청에서 수집된 지표

메트릭 이름	설명	형식
ApiCallDuration	API 호출 지속 시간입니다. 여기에는 모든 호출 시도가 포함됩니다.	지속 시간*
ApiCallSuccessful	API 호출이 성공하면 true이고, 그렇지 않으면 false입니다.	불
CredentialsFetchDuration	API 호출용 서명 자격 증명을 가져오는 지속 시간입니다.	지속 시간*
EndpointResolveDuration	API 호출에 사용되는 엔드포인트를 해결하는 데 걸리는 시간입니다.	지속 시간*
MarshallingDuration	SDK 요청을 HTTP 요청으로 마샬하는 데 걸리는 시간입니다.	지속 시간*
OperationName	호출 중인 서비스 작업의 이름입니다.	String
RetryCount	SDK가 요청 실행 시 수행한 재시도 횟수입니다. 0은 요청이	Integer

메트릭 이름	설명	형식
	처음 작동했고 재시도가 시도되지 않았음을 의미합니다. 재시도 동작 구성에 대한 자세한 내용은 재시도 전략 섹션을 참조하세요.	
ServiceId	서비스의 고유 ID입니다.	String
ServiceEndpoint	서비스의 엔드포인트입니다.	URI
TokenFetchDuration	API 호출용 서명 자격 증명을 가져오는 지속 시간입니다.	지속 시간*

*java.time.Duration.

각 요청 시도에 대해 수집된 지표

응답을 수신하려면 각 API 호출에 여러 번 시도해야 할 수 있습니다. 이 지표는 각 요청 시도에 대해 수집됩니다.

주요 지표

메트릭 이름	설명	형식
AwsExtendedRequestId	서비스 요청의 확장 요청 ID입니다.	String
AwsRequestId	서비스 요청의 요청 ID입니다.	String
BackoffDelayDuration	이 API 호출을 시도하기 전에 SDK가 대기한 지속 시간입니다. 값은 클라이언트에 설정된 `BackoffStrategy` 를 기반으로 합니다. 자세한 내용은 이 가이드의 재시도 전략 섹션을 참조하세요.	지속 시간*

메트릭 이름	설명	형식
ErrorType	호출 시도에서 발생한 오류 유형입니다. 유효한 값은 다음과 같습니다. • `Throttling` : 서비스가 스로틀링 오류로 응답했습니다. • `ServerError` : 서비스가 스로틀링 이외의 오류로 응답했습니다. • `ConfiguredTimeout` : API 호출 수준 또는 API 호출 시도 수준에서 클라이언트 제한 시간이 발생했습니다. • `IO`: I/O 오류가 발생했습니다. • `Other`: 위의 범주 목록 중 하나에 속하지 않는 다른 오류의 경우 Catch-all입니다.	String

메트릭 이름	설명	형식
ReadThroughput	`NumberOfResponseBytesRead / (TTLB - TTFB)`로 정의된 클라이언트의 읽기 처리량입니다. 이 값은 초당 바이트 단위입니다. 이 지표는 `ResponseTransformer` 또는 `AsyncResponseTransformer` 내에서 읽은 바이트만 측정합니다. 응답 스트림이 변환기의 결과로 반환되는 경우와 같이 변환기 외부에서 읽는 데이터는 계산에 포함되지 않습니다.	Double
ServiceCallDuration	서비스에 연결하고(또는 연결 풀에서 연결을 획득하고), 직렬화된 요청을 보내고, 초기 응답(예: HTTP 상태 코드 및 헤더)을 수신하는 지속 시간입니다. 여기에는 서비스의 전체 응답을 읽는 시간은 포함되지 않습니다.	지속 시간*
SigningDuration	HTTP 요청에 서명하는 지속 시간입니다.	지속 시간*
TimeToFirstByte	서비스에 HTTP 요청(연결 획득 포함)을 보내고 응답에서 헤더의 첫 번째 바이트를 수신하는 지속 시간입니다.	지속 시간*

메트릭 이름	설명	형식
TimeToLastByte	서비스에 HTTP 요청(연결 획득 포함)을 보내고 응답에서 마지막 바이트를 수신하는 지속 시간입니다. 스트리밍 응답을 반환하는 API의 경우 이 지표는 `ResponseTransformer` 또는 `AsyncResponseTransformer` 가 완료될 때까지의 시간에 걸쳐 있습니다.	지속 시간*
UnmarshallingDuration	SDK 응답에 대한 HTTP 응답을 언마샬하는 데 걸리는 지속 시간입니다. 참고: 스트리밍 작업의 경우 응답 페이로드를 읽는 데 걸리는 시간은 포함되지 않습니다.	지속 시간*

*java.time.Duration.

HTTP 지표

메트릭 이름	설명	형식	HTTP 클라이언트 필요*
AvailableConcurrency	대상 서버에 대한 새 연결을 설정하지 않고 HTTP 클라이언트가 지원하는 추가 동시 요청 수입니다. HTTP/1 작업의 경우 서비스에 설정된 유휴 TCP 연결 수와 같습니다. HTTP/2 작업의 경우 유휴 스트림 수와 같습니다. 참고: 이 값은 HTTP 클라이언트 구현에 따라 다릅니다.	Integer	Apache, Netty, CRT

메트릭 이름	설명	형식	HTTP 클라이언트 필요*
	• Apache 클라이언트: 값이 전체 HTTP 클라이언트에 적용됨 • Netty 클라이언트: 엔드포인트당 값이 적용됨 • AWS CRT 기반 클라이언트: 엔드포인트당 값이 적용됨 값은 개별 HTTP 클라이언트 인스턴스로 범위가 지정되며 동일한 JVM의 다른 HTTP 클라이언트에서 동시성을 제외합니다.		
ConcurrencyAcquireDuration	연결 풀에서 채널을 획득하는 데 걸린 지속 시간입니다. HTTP/1 작업의 경우 채널은 TCP 연결과 같습니다. HTTP/2 작업의 경우 채널은 HTTP/2 스트림 채널과 같습니다. 새 채널을 획득하는 데 다음과 같은 시간이 포함될 수 있습니다. 1. 클라이언트의 최대 동시성 구성으로 제한되는 동시성 허용 대기 시간입니다. 2. 풀에서 기존 연결을 사용할 수 없는 경우 새 연결을 설정합니다. 3. TLS가 사용 설정된 경우 TLS 핸드셰이크 및 협상을 수행합니다.	지속 시간*	Apache, Netty, CRT
HttpClientName	요청에 사용 중인 HTTP의 이름입니다.	String	Apache, Netty, CRT
HttpStatusCode	HTTP 응답의 상태 코드입니다.	Integer	임의

메트릭 이름	설명	형식	HTTP 클라이언트 필요*
LeasedConcurrency	HTTP 클라이언트에서 실행하는 요청 수입니다. HTTP/1 작업의 경우 서비스와의 활성 TCP 연결 수와 같습니다(유휴 연결 제외). HTTP/2 작업의 경우 서비스가 포함된 활성 HTTP 스트림 수와 같습니다(유휴 스트림 용량 제외). 참고: 이 값은 HTTP 클라이언트 구현에 따라 다릅니다. • Apache 클라이언트: 값이 전체 HTTP 클라이언트에 적용됨 • Netty 클라이언트: 엔드포인트당 값이 적용됨 • AWS CRT 기반 클라이언트: 엔드포인트당 값이 적용됨 값은 개별 HTTP 클라이언트 인스턴스로 범위가 지정되며 동일한 JVM의 다른 HTTP 클라이언트에서 동시성을 제외합니다.	Integer	Apache, Netty, CRT
LocalStreamWindowSize	이 요청이 실행되는 스트림의 로컬 HTTP/2 창 크기(바이트)입니다.	Integer	Netty

메트릭 이름	설명	형식	HTTP 클라이언트 필요*
MaxConcurrency	HTTP 클라이언트가 지원하는 최대 동시 요청 수입니다. HTTP/1 작업의 경우 HTTP 클라이언트가 풀링할 수 있는 최대 TCP 연결 수와 같습니다. HTTP/2 작업의 경우 HTTP 클라이언트가 풀링할 수 있는 최대 스트림 수와 같습니다. 참고: 이 값은 HTTP 클라이언트 구현에 따라 다릅니다. • Apache 클라이언트: 값이 전체 HTTP 클라이언트에 적용됨 • Netty 클라이언트: 엔드포인트당 값이 적용됨 • AWS CRT 기반 클라이언트: 엔드포인트당 값이 적용됨 값은 개별 HTTP 클라이언트 인스턴스로 범위가 지정되며 동일한 JVM의 다른 HTTP 클라이언트에서 동시성을 제외합니다.	Integer	Apache, Netty, CRT

메트릭 이름	설명	형식	HTTP 클라이언트 필요*
PendingConcurrencyAcquires	HTTP 클라이언트의 동시성을 기다리는 요청 수입니다. HTTP/1 작업의 경우 TCP 연결이 연결 풀에서 설정되거나 반환될 때까지 대기하는 요청 수와 같습니다. HTTP/2 작업의 경우 연결 풀에서 새 스트림(및 가능한 경우 새 HTTP/2 연결)을 기다리는 요청 수와 같습니다. 참고: 이 값은 HTTP 클라이언트 구현에 따라 다릅니다. • Apache 클라이언트: 값이 전체 HTTP 클라이언트에 적용됨 • Netty 클라이언트: 엔드포인트당 값이 적용됨 • AWS CRT 기반 클라이언트: 엔드포인트당 값이 적용됨 값은 개별 HTTP 클라이언트 인스턴스로 범위가 지정되며 동일한 JVM의 다른 HTTP 클라이언트에서 동시성을 제외합니다.	Integer	Apache, Netty, CRT
RemoteStreamWindowSize	이 요청이 실행되는 스트림의 원격 HTTP/2 창 크기(바이트)입니다.	Integer	Netty

*java.time.Duration.

열에 사용된 용어의 의미는 다음과 같습니다.

- Apache: Apache 기반 HTTP 클라이언트(`ApacheHttpClient`)
- Netty: Netty 기반 HTTP 클라이언트(`NettyNioAsyncHttpClient`)
- CRT: AWS CRT 기반 HTTP 클라이언트(`AwsCrtAsyncHttpClient`)
- 임의: 지표 데이터 수집은 HTTP 클라이언트에 종속되지 않습니다. 여기에는 URLConnection 기반 HTTP 클라이언트(`UrlConnectionHttpClient`)가 포함됨

AWS SDK for Java 2.x 애플리케이션 모니터링

AWS SDK for Java 2.x를 사용하는 애플리케이션의 신뢰성, 가용성 및 성능을 유지하려면 모니터링이 중요합니다. AWS는 SDK for Java 2.x를 모니터링하고, 이상이 있을 때 이를 보고하고, 필요한 경우 자동 조치를 취할 수 있도록 다음과 같은 모니터링 도구를 제공합니다.

- Amazon CloudWatch는 AWS에서 실행하는 AWS 리소스와 애플리케이션을 실시간으로 모니터링합니다. 지표를 수집 및 추적하고, 사용자 지정 대시보드를 생성할 수 있으며, 지정된 지표가 지정한 임곗값에 도달하면 사용자에게 알리거나 조치를 취하도록 경보를 설정할 수 있습니다. 예를 들어 CloudWatch에서 Amazon EC2 인스턴스의 CPU 사용량 또는 기타 지표를 추적하고 필요할 때 자동으로 새 인스턴스를 시작할 수 있습니다. 자세한 내용은 CloudWatch 사용 설명서를 참조하세요.
- Amazon CloudWatch Logs로 Amazon EC2 인스턴스, CloudTrail, 기타 소스의 로그 파일을 모니터링, 저장 및 액세스할 수 있습니다. CloudWatch Logs는 로그 파일의 정보를 모니터링하고 특정 임곗값에 도달하면 사용자에게 알릴 수 있습니다. 또한 매우 내구력 있는 스토리지에 로그 데이터를 저장할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs 사용 설명서를 참조하세요.
- AWS CloudTrail은 직접 수행하거나 AWS 계정을 대신하여 수행한 API 직접 호출 및 관련 이벤트를 캡처하고 지정한 Amazon S3 버킷에 로그 파일을 전송합니다. 어떤 사용자 및 계정이 AWS를 직접적으로 호출했는지, 어떤 소스 IP 주소에 직접 호출이 이루어졌는지, 언제 직접 호출이 발생했는지 확인할 수 있습니다. 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.

SDK를 사용하여 Java 2.x로 로깅하기

AWS SDK for Java 2.x는 SLF4J(런타임에 여러 로깅 시스템 중 하나를 활성화하는 추상 계층)을 사용합니다.

지원되는 로깅 시스템에는 Java Logging Framework와 Apache Log4j 2 등이 있습니다. 이 항목에서는 Log4j 2를 SDK 작업을 위한 로깅 시스템으로 사용하는 방법을 보여줍니다.

Log4j 구성 파일

일반적으로 Log4j 2을 포함하고 이름이 `log4j2.xml`로 지정된 구성 파일을 사용합니다. 아래에는 예제 구성 파일이 나와 있습니다. 구성 파일에 사용된 값에 대한 자세한 내용은 Log4j 구성에 대한 설명서를 참조하세요.

애플리케이션을 시작할 때 `log4j2.xml` 파일이 클래스 경로에 있어야 합니다. Maven 프로젝트의 경우 파일을 `<project-dir>/src/main/resources` 디렉터리에 넣으세요.

`log4j2.xml` 구성 파일은 로깅 출력이 전송될 대상인 로깅 수준(예를 들면, 파일 또는 콘솔), 출력 형식 같은 속성을 지정합니다. 로깅 수준은 Log4j 2가 출력하는 세부 수준을 지정합니다. Log4j는 여러 로깅 계층의 개념을 지원합니다. 로깅 수준은 각 계층마다 독립적으로 설정됩니다. AWS SDK for Java 2.x와 함께 사용하는 기본 로깅 계층은 `software.amazon.awssdk`입니다.

로깅 종속성 추가

빌드 파일에서 SLF4J Log4j 2 바인딩을 구성하려면 다음을 사용하세요.

Maven

다음 요소를 `pom.xml` 파일에 추가합니다.

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

Gradle–Kotlin DSL

다음을 `build.gradle.kts` 파일에 추가합니다.

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

`log4j-slf4j2-impl` 아티팩트의 최소 버전에 `2.20.0`를 사용 최신 버전의 경우 Maven Central에 게시된 버전을 사용하세요. *VERSION*을 사용할 버전으로 교체하세요.

서비스 관련 오류 및 경고

SDK 클라이언트 라이브러리에서 중요한 메시지를 포착하려면 항상 "software.amazon.awssdk" 로거 계층 구조를 "WARN"으로 설정해 두는 것이 좋습니다. 예를 들어 Amazon S3 클라이언트가 애플리케이션이 `InputStream`를 제대로 닫지 않았고 리소스가 누출될 수 있음을 감지하면 S3 클라이언트는

경고 메시지를 통해 이를 로그에 보고합니다. 또한 클라이언트에 요청 또는 응답 처리 문제가 발생하는 경우에도 메시지가 기록됩니다.

다음 `log4j2.xml` 파일은 `rootLogger`를 “WARN”으로 설정합니다. 그러면 “software.amazon.awssdk” 계층 구조에 있는 로거를 포함하여 애플리케이션에 있는 모든 로거에서 경고 및 오류 수준 메시지가 출력됩니다. 또는 `<Root level="ERROR">`를 사용하는 경우 “software.amazon.awssdk” 로거 계층을 명시적으로 “WARN”으로 설정할 수도 있습니다.

예제 Log4j2.xml 구성 파일

이 구성은 모든 로거 계층에 대해 “ERROR” 및 “WARN” 수준의 메시지를 콘솔에 기록합니다.

```
<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>
```

요청 및 응답 요약 로깅

AWS 서비스를 요청할 때마다 고유한 AWS 요청 ID가 생성되는데, 이는 AWS 서비스가 요청을 처리하는 방식에 문제가 발생할 경우 유용합니다. AWS 요청 ID는 실패한 서비스 호출에 대해 SDK의 `SdkServiceException` 객체를 통해 프로그래밍 방식으로 액세스할 수 있으며, “software.amazon.awssdk.request” 로거의 DEBUG 로그 수준을 통해 보고할 수도 있습니다.

다음 `log4j2.xml` 파일은 요청 및 응답의 요약을 제공합니다.

```
<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>
```

다음은 로그 출력의 예입니다:

```
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
```

요청 ID에만 관심이 있는 경우 `<Logger name="software.amazon.awssdk.requestId" level="DEBUG" />`를 사용하세요.

디버그 수준 SDK 로깅

SDK가 수행하는 작업에 대한 자세한 정보가 필요한 경우 `software.amazon.awssdk` 로거의 로깅 수준을 `DEBUG`로 설정할 수 있습니다. 이 수준에서 SDK는 많은 세부 정보를 출력하므로 통합 테스트를 사용하여 오류를 해결하려면 이 수준을 설정하는 것이 좋습니다.

이 로깅 수준에서 SDK는 구성, 자격 증명 해결, 실행 인터셉터, 개괄적인 TLS 활동, 요청 서명 등에 대한 정보를 로깅합니다.

다음은 `S3Client#listBuckets()` 호출에 대해 SDK가 `DEBUG` 수준에서 출력하는 문의 샘플입니다.

```
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
...
```

다음 `log4j2.xml` 파일은 이전 출력을 구성합니다.

```
<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>
```

유선 로깅 사용

Java 2.x용 SDK가 보내고 받는 정확한 요청과 응답을 확인하는 것이 유용할 수 있습니다. 이 정보에 액세스해야 하는 경우 서비스 클라이언트가 사용하는 HTTP 클라이언트에 따라 필요한 구성을 추가하여 일시적으로 활성화할 수 있습니다.

기본적으로 S3Client와 같은 동기 서비스 클라이언트는 기본 Apache HTTPClient를 사용하고 S3AsyncClient와 같은 비동기 서비스 클라이언트는 Netty 비차단 HTTP 클라이언트를 사용합니다.

다음은 두 가지 범주의 서비스 클라이언트에 사용할 수 있는 HTTP 클라이언트의 분석입니다.

동기식 HTTP 클라이언트	동기식 HTTP 클라이언트
ApacheHttpClient(기본값)	NettyNioAsyncHttpClient(기본값)
UrlConnectionHttpClient	AwsCrtAsyncHttpClient
AwsCrtHttpClient	

기본 HTTP 클라이언트에 따라 추가해야 하는 구성 설정은 아래 해당 탭을 참조하세요.

> **Warning**
>
> 유선 로깅은 디버깅 목적에만 사용하는 것이 좋습니다. 로그에 민감한 데이터가 될 수 있기 때문에 프로덕션 환경에서는 비활성화합니다. HTTPS 호출을 포함, 암호화 없는 전체 요청이나 응답을 로그로 기록합니다. 대량 요청(예: Amazon S3로 파일 업로드)이나 응답의 경우 verbose 유선 로깅이 애플리케이션 성능에 큰 영향을 줄 수도 있습니다.

ApacheHttpClient

`log4j2.xml` 구성 파일에 “org.apache.http.wire” 로거를 추가하고 레벨을 “DEBUG”로 설정합니다.

다음 `log4j2.xml` 파일은 Apache HttpClient에서 전체 유선 로깅을 활성화시킵니다.

```
<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>
```

Apache는 내부적으로 1.2를 사용하기 때문에 Apache를 통한 유선 로깅에는 `log4j-1.2-api` 아티팩트에 대한 추가 Maven 종속성이 필요합니다.

Apache HTTP 클라이언트의 유선 로깅을 포함하여 log4j 2의 전체 Maven 종속성 집합은 다음 빌드 파일 코드 조각에 나와 있습니다.

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")
}
...
```

`log4j-bom` 아티팩트의 최소 버전에 `2.20.0`를 사용 최신 버전의 경우 Maven Central에 게시된 버전을 사용하세요. *VERSION*을 사용할 버전으로 교체하세요.

UrlConnectionHttpClient

`UrlConnectionHttpClient`를 사용하는 서비스 클라이언트의 세부 정보를 기록하려면 먼저 다음 내용이 포함된 `logging.properties` 파일을 만드세요.

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

`logging.properties`의 전체 경로를 사용하여 다음 JVM 시스템 속성을 설정합니다.

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

이 구성은 요청 및 응답의 헤더만 기록합니다. 예를 들면 다음과 같습니다.

```
<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}
```

요청 alc 응답 본문을 보려면 JVM 속성에 `-Djavax.net.debug=all`를 추가하세요. 이 추가 속성은 모든 SSL 정보를 포함하여 많은 양의 정보를 기록합니다.

로그 콘솔 또는 로그 파일 내에서 실제 요청 및 응답이 포함된 로그 단원으로 빠르게 이동하려면 `"GET"`나 `"POST"`를 검색하세요. 요청으로는 `"Plaintext before ENCRYPTION"`을 검색하고 응답으로는 `"Plaintext after DECRYPTION"` 검색하면 헤더와 본문의 전체 텍스트를 볼 수 있습니다.

NettyNioAsyncHttpClient

비동기 서비스 클라이언트가 기본값 `NettyNioAsyncHttpClient`을 사용하는 경우 HTTP 헤더와 요청 및 응답 본문에 대한 `log4j2.xml` 파일에 로거 2개를 추가하세요.

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

다음은 전체 `log4j2.xml` 예제입니다.

```
<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>
```

이러한 설정은 모든 헤더 세부 정보와 요청 및 응답 본문을 기록합니다.

AwsCrtAsyncHttpClient/AwsCrtHttpClient

AWS CRT 기반 HTTP 클라이언트의 인스턴스를 사용하는 서비스 클라이언트를 구성한 경우 JVM 시스템 속성을 설정하거나 프로그래밍 방식으로 세부 정보를 로그할 수 있습니다.

Log to a file at "Debug" level

시스템 속성 사용:

```
-Daws.crt.log.leve
l=Trace
-Daws.crt.log.des
tination=File
-Daws.crt.log.fil
ename=<path to file>
```

프로그래밍 방식으로

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

// Execute this statement before constructing the
 SDK service client.
Log.initLoggingToFile(Log.LogLevel.Trace,
 "<path to file>");
```

Log to the console at "Debug" level

시스템 속성 사용:

```
-Daws.crt.log.leve
l=Trace
-Daws.crt.log.des
tination=Stdout
```

프로그래밍 방식으로

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

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

보안상의 이유로 ‘Trace’ 수준에서 AWS CRT 기반 HTTP 클라이언트는 응답 헤더만 로그합니다. 요청 헤더, 요청 본문 및 응답 본문은 로깅되지 않습니다.

AWS SDK for Java 2.x에서 클라이언트 엔드포인트 구성

SDK for Java 2.x는 서비스 엔드포인트를 구성하는 다양한 방법을 제공합니다. 엔드포인트는 SDK가 AWS 서비스에 대한 API 호출을 만드는 데 사용되는 URL입니다. 기본적으로 SDK는 구성한 AWS 리전에 따라 각 서비스에 적절한 엔드포인트를 자동으로 결정합니다. 그러나 이러한 엔드포인트를 사용자 지정하거나 재정의해야 하는 시나리오가 있습니다.

- 로컬 또는 서드 파티 서비스 구현 작업(예: LocalStack)
- 프록시 또는 VPC 엔드포인트를 통해 AWS 서비스에 연결
- 베타 또는 시험판 서비스 엔드포인트에 대한 테스트

엔드포인트 구성 옵션

AWS SDK for Java 2.x는 엔드포인트를 구성하는 몇 가지 방법을 제공합니다.

- 서비스 클라이언트 빌더를 사용한 코드 내 구성
- 환경 변수를 사용한 외부 구성
- JVM 시스템 속성을 사용한 외부 구성
- 공유 AWS 구성 파일을 사용한 외부 구성

코드 내 엔드포인트 구성

`endpointOverride` 사용하기

엔드포인트를 구성하는 가장 직접적인 방법은 서비스 클라이언트 빌더에서 `endpointOverride` 메서드를 사용하는 것입니다. 이 메서드는 사용자 지정 엔드포인트 URL을 나타내는 `URI` 객체를 허용합니다.

Example Amazon S3 클라이언트에 대한 사용자 지정 엔드포인트 설정

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import java.net.URI;

S3Client s3 = S3Client.builder()
```

```
        .region(Region.US_WEST_2)
        .endpointOverride(URI.create("https://my-custom-s3-endpoint.example.com"))
        .build();
```

endpointOverride를 사용할 때 엔드포인트가 명시적으로 설정되더라도 클라이언트의 리전을 지정해야 합니다. 리전은 요청에 서명하는 데 사용됩니다.

엔드포인트 검색

일부 AWS 서비스는 SDK가 사용할 최적의 엔드포인트를 자동으로 검색할 수 있는 엔드포인트 검색을 지원합니다. 서비스 클라이언트 빌더에서 endpointDiscoveryEnabled 메서드를 사용하여 이 기능을 사용하거나 사용 해제할 수 있습니다.

Example DynamoDB 클라이언트에 대한 엔드포인트 검색 사용

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;

DynamoDbClient dynamoDb = DynamoDbClient.builder()
        .region(Region.US_WEST_2)
        .endpointDiscoveryEnabled(true)
        .build();
```

요청 수준 엔드포인트 구성

경우에 따라 기본 엔드포인트가 있는 다른 요청에 동일한 클라이언트를 사용하는 동안 특정 요청에 대해 엔드포인트를 재정의해야 할 수 있습니다. AWS SDK for Java 2.x는 요청 재정의를 통해 이를 지원합니다.

Example특정 요청에 대한 엔드포인트 재정의

```
import software.amazon.awssdk.core.SdkRequest;
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.http.SdkHttpRequest;
```

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

// Create a request
GetObjectRequest getObjectRequest = GetObjectRequest.builder()
        .bucket("amzn-s3-demo-bucket")
        .key("my-key")
        .overrideConfiguration(c -> c.putHeader("Host", "custom-endpoint.example.com"))
        .build();

// Execute the request with the custom endpoint
s3.getObject(getObjectRequest);
```

요청 수준 엔드포인트 재정의는 제한되며 일부 서비스 또는 시나리오에서는 작동하지 않을 수 있습니다. 대부분의 경우 클라이언트 수준 엔드포인트 구성을 사용하는 것이 좋습니다.

외부 엔드포인트 구성

환경 변수 사용

환경 변수를 사용하여 엔드포인트를 구성합니다. SDK는 `AWS_ENDPOINT_URL_[SERVICE]` 형식의 환경 변수를 통해 서비스별 엔드포인트 구성을 지원하며, 여기서 `[SERVICE]`는 대문자 서비스 식별자입니다.

Example환경 변수를 사용한 S3 엔드포인트 설정

```
# For Linux/macOS
export AWS_ENDPOINT_URL_S3=https://my-custom-s3-endpoint.example.com

# For Windows
set AWS_ENDPOINT_URL_S3=https://my-custom-s3-endpoint.example.com
```

또는 다음 환경 변수를 사용하여 글로벌 엔드포인트 URL 접두사 또는 접미사를 설정할 수 있습니다.

- `AWS_ENDPOINT_URL` - 모든 서비스에 대한 글로벌 엔드포인트를 설정합니다.
- `AWS_ENDPOINT_URL_PREFIX` - 모든 서비스 엔드포인트에 접두사를 추가합니다.

- `AWS_ENDPOINT_URL_SUFFIX` - 모든 서비스 엔드포인트에 접미사를 추가합니다.

JVM 시스템 속성 사용

또한 JVM 시스템 속성을 사용하여 엔드포인트를 구성할 수 있습니다. 형식은 환경 변수와 비슷하지만 다른 이름 지정 규칙을 사용합니다.

Example JVM 시스템 속성을 사용하여 S3 엔드포인트 설정

```
java -Daws.endpointUrl.s3=https://my-custom-s3-endpoint.example.com -jar your-
application.jar
```

또한 글로벌 엔드포인트 구성은 시스템 속성을 통해서 사용할 수 있습니다.

- `aws.endpointUrl` - 모든 서비스에 대한 글로벌 엔드포인트를 설정합니다.
- `aws.endpointUrl.prefix` - 모든 서비스 엔드포인트에 접두사를 추가합니다.
- `aws.endpointUrl.suffix` - 모든 서비스 엔드포인트에 접미사를 추가합니다.

공유 AWS 구성 파일 사용

또한 AWS SDK for Java 2.x는 일반적으로 `~/.aws/config`(Linux/macOS) 또는 `%USERPROFILE%\.aws\config`(Windows)에 있는 공유 AWS 구성 파일을 통해 엔드포인트 구성을 지원합니다. 자세한 내용과 예제는 AWS SDK 및 도구 참조 안내서를 참조하세요.

구성 우선 순위

여러 엔드포인트 구성이 있는 경우 SDK는 다음 우선순위(가장 높음에서 가장 낮음)를 따릅니다.

1. 요청 수준 재정의(해당하는 경우)
2. `endpointOverride`를 통한 클라이언트 수준 구성
3. 환경 변수
4. JVM 시스템 속성
5. 공유 AWS 구성 파일
6. 구성된 AWS 리전을 기반으로 하는 기본 엔드포인트

서비스별 엔드포인트 구성

일부 AWS 서비스에는 해당 서비스와 관련된 추가 엔드포인트 구성 옵션이 있습니다. 다음은 몇 가지 예입니다.

Amazon S3 엔드포인트 구성

Amazon S3는 `S3Configuration` 클래스를 통해 여러 엔드포인트 구성을 지원합니다.

- `dualstackEnabled` - IPv6 지원 사용
- `accelerateModeEnabled` - S3 Transfer Acceleration 사용
- `pathStyleAccessEnabled` - 가상 호스팅식 대신 경로식 액세스 사용
- `useArnRegionEnabled` - 크로스 리전 요청에 ARN의 리전 사용
- `fipsModeEnabled` - 요청을 FIPS 준수 엔드포인트로 라우팅

Example S3별 엔드포인트 옵션 구성

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.S3Configuration;

S3Client s3 = S3Client.builder()
        .region(Region.US_WEST_2)
        .serviceConfiguration(S3Configuration.builder()
                .accelerateModeEnabled(true)
                .dualstackEnabled(true)
                .pathStyleAccessEnabled(false)
                .fipsModeEnabled(true)
                .build())
        .build();
```

DynamoDB 엔드포인트 구성

DynamoDB의 경우 엔드포인트 검색을 사용하거나 DynamoDB 로컬에 연결하여 테스트할 수 있습니다.

Example DynamoDB 로컬에 연결

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import java.net.URI;

DynamoDbClient dynamoDb = DynamoDbClient.builder()
        .endpointOverride(URI.create("http://localhost:8000"))
        // The region is meaningless for DynamoDB local but required for the client
 builder.
        .region(Region.US_WEST_2)
        .build();
```

또한 DynamoDB는 코드에서 구성하거나 외부 설정을 사용하여 구성할 수 있는 계정 기반 엔드포인트 사용을 지원합니다. 다음 예제에서는 클라이언트를 만들 때 코드에서 계정 기반 엔드포인트 사용을 사용 해제하는 방법을 보여줍니다(기본 설정: 선호).

```
DynamoDbClient dynamoDbClient = DynamoDbClient.builder()
    .region(Region.US_EAST_1)
    .accountIdEndpointMode(AccountIdEndpointMode.DISABLED)
    .build();
```

모범 사례

AWS SDK for Java 2.x에서 엔드포인트를 구성할 때 다음 모범 사례를 고려하세요.

- 환경별 엔드포인트에 외부 구성 사용 - 환경 변수, 시스템 속성 또는 환경(개발, 테스트, 프로덕션) 간에 다른 엔드포인트에 AWS 구성 파일을 사용합니다.
- 애플리케이션별 엔드포인트에 코드 내 구성 사용 - 애플리케이션의 설계에 맞는 엔드포인트에 클라이언트 빌더의 `endpointOverride` 메서드를 사용합니다.
- 항상 리전 지정 - 엔드포인트를 재정의할 때도 항상 요청 서명에 사용되는 리전을 지정합니다.
- 글로벌 엔드포인트 재정의에 주의 - 글로벌 엔드포인트 재정의는 모든 서비스에 영향을 미칠 수 있으며, 이는 의도한 것과 다를 수 있습니다.
- 보안 영향 고려 - 사용자 지정 엔드포인트를 사용할 경우 특히 프로덕션 워크로드에 적절한 보안 조치가 있는지 확인합니다.

에서 HTTP 클라이언트 구성 AWS SDK for Java 2.x

이 섹션에서는 서비스 클라이언트에 사용할 HTTP 클라이언트를 변경하고 AWS SDK for Java 2.x를 사용하여 HTTP 클라이언트의 기본 구성을 변경할 수 있습니다. 이 단원에서는 SDK의 HTTP 클라이언트 및 설정에 대해 설명합니다.

SDK for Java에서 사용 가능한 HTTP 클라이언트

동기식 HTTP 클라이언트

SDK for Java의 동기식 HTTP 클라이언트는 SdkHttpClient 인터페이스를 구현합니다. 동기식 서비스 클라이언트(예: `S3Client` 또는 `DynamoDbClient`)를 사용하려면 동기식 HTTP 클라이언트를 사용해야 합니다. 는 세 개의 동기 HTTP 클라이언트를 AWS SDK for Java 제공합니다.

ApacheHttpClient(기본값)

ApacheHttpClient는 동기 서비스 클라이언트를 위한 기본 HTTP 클라이언트입니다. `ApacheHttpClient` 구성에 대한 내용은 Apache 기반 HTTP 클라이언트 설정을 참조하세요.

AwsCrtHttpClient

AwsCrtHttpClient는 높은 처리량과 비차단 IO를 제공합니다. 공통 AWS 런타임(CRT) Http 클라이언트를 기반으로 합니다. `AwsCrtHttpClient`를 구성하고 서비스 클라이언트와 함께 사용하는 방법에 대한 자세한 내용은 the section called “AWS CRT 기반 HTTP 클라이언트 구성”을 참조하세요.

UrlConnectionHttpClient

애플리케이션에서 사용하는 항아리 및 타사 라이브러리 수를 최소화하려면 UrlConnectionHttpClient를 사용하면 됩니다. `UrlConnectionHttpClient` 구성에 대한 내용은 URL 연결 기반 HTTP 클라이언트 구성을 참조하세요.

Apache5HttpClient

Apache5HttpClient는 Apache 5.x HttpClient를 기반으로 구축`ApacheHttpClient`된의 업데이트된 버전입니다. HttpClient 버전 5.x는 Apache에서 적극적으로 유지 관리하는 버전이며 Java 21에 대한 가상 스레드 지원, SLF4J를 통한 향상된 로깅 유연성 등 최신 Java 에코시스템 호환성을 `ApacheHttpClient` 제공하여에서 사용하는 이전 버전을 개선합니다. `Apache5HttpClient`는 향후 버전의 Java 2.x용 AWS SDK에서를 기본 동기식 클라이언트`ApacheHttpClient`로 대체합니다. API와 기능이 동일하므로 드롭인 대체로 적합합니다. `Apache5HttpClient` 구성에 대한 내용은 Apache 5.x 기반 HTTP 클라이언트 구성을 참조하세요.

비동기 클라이언트

SDK for Java의 비동기 HTTP 클라이언트는 SdkAsyncHttpClient 인터페이스를 구현합니다. 비동기 서비스 클라이언트(예: `S3AsyncClient` 또는 `DynamoDbAsyncClient`)를 사용하려면 비동기 HTTP 클라이언트를 사용해야 합니다. 는 두 개의 비동기 HTTP 클라이언트를 AWS SDK for Java 제공합니다.

NettyNioAsyncHttpClient(기본값)

NettynioAsynchTPClient는 비동기 클라이언트가 사용하는 기본 HTTP 클라이언트입니다. `NettyNioAsyncHttpClient` 구성에 대한 내용은 the section called “Netty 기반 HTTP 클라이언트를 구성”을 참조하세요.

AwsCrtAsyncHttpClient

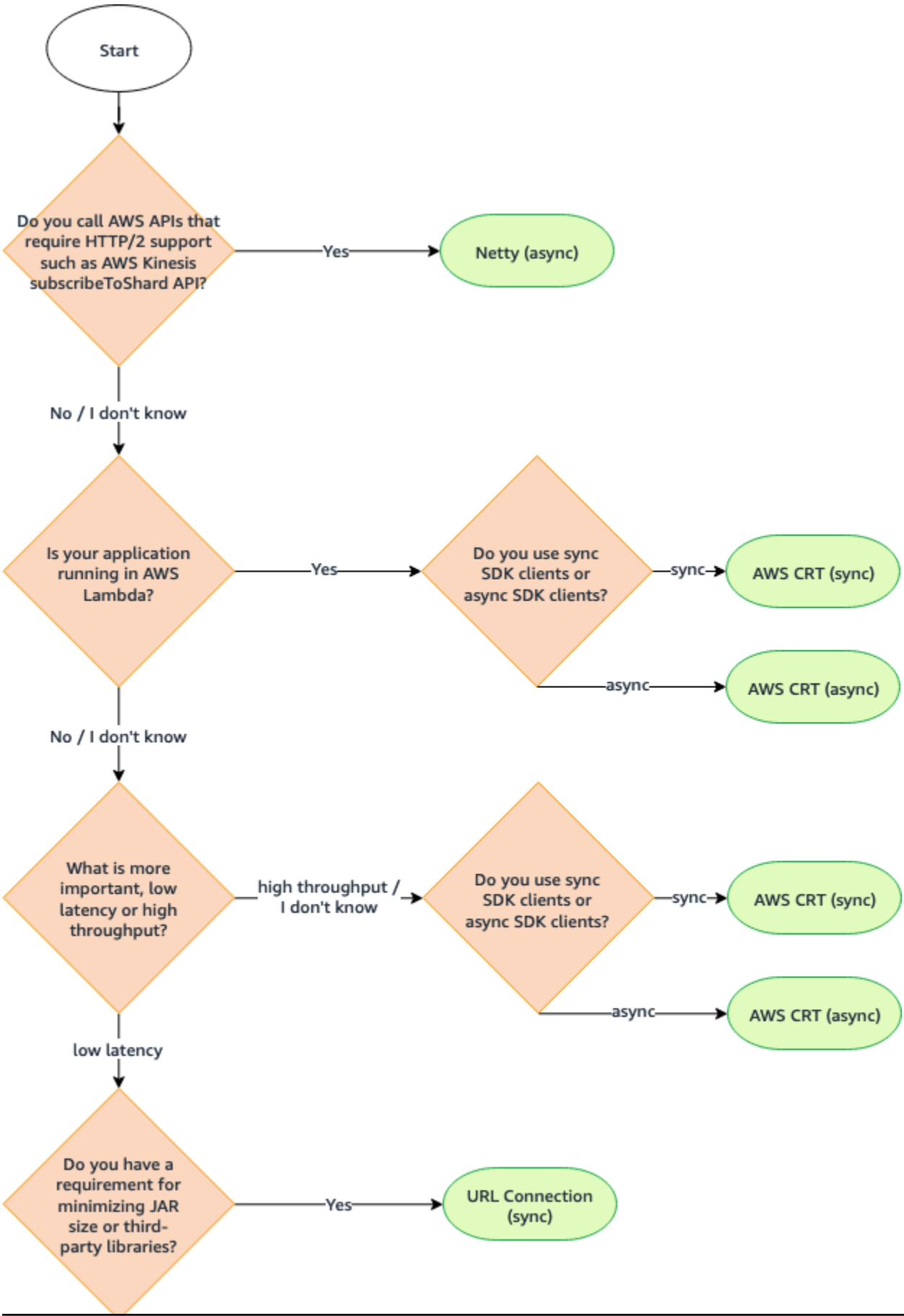
AwsCrtAsyncHttpClient는 AWS 공통 런타임(CRT) HTTP 클라이언트를 기반으로 합니다. `AwsCrtAsyncHttpClient` 구성에 대한 내용은 the section called “AWS CRT 기반 HTTP 클라이언트 구성”을 참조하세요.

HTTP 클라이언트 권장 사항

HTTP 클라이언트 구현을 선택할 때는 몇 가지 요인이 작용합니다. 다음 정보는 결정을 하는 데 도움을 줍니다.

권장 사항 순서도

다음 순서도는 사용할 HTTP 클라이언트를 결정하는 데 도움이 되는 일반적인 지침을 제공합니다.


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 클라이언트 비교

다음 표에는 각 HTTP 클라이언트에 대한 자세한 정보가 나와 있습니다.

HTTP 클라이언트	동기 또는 비동기	사용해야 하는 경우	제한 및 단점
Apache 기반 HTTP 클라이언트 (기본 동기화 HTTP 클라이언트)	동기화	높은 처리량보다 낮은 지연 시간을 선호하는 경우 사용	다른 HTTP 클라이언트에 비해 시작 시간이 느림
URLConnection 기반 HTTP 클라이언트	동기화	타사 종속성을 제한해야 하는 까다로운 요구 사항이 있는 경우 사용	Amazon APIGateway Update 작업과 같은 일부 API에 필요한 HTTP PATCH 메서드를 지원하지 않
AWS CRT 기반 동기화 HTTP 클라이언트[1]	동기화	• 애플리케이션이에서 실행 중인 경우 사용 AWS Lambda • 짧은 지연 시간보다 높은 처리량을 선호하는 경우 사용 • SDK 클라이언트 동기화를 선호하는 경우 사용	다음 Java 시스템 속성은 지원되지 않습니다. • javax.net.ssl.keyStore • javax.net.ssl.keyStorePassword • javax.net.ssl.trustStore • javax.net.ssl.trus

HTTP 클라이언트	동기 또는 비동기	사용해야 하는 경우	제한 및 단점
			tStorePassword
Netty 기반 HTTP 클라이언트 (기본 비동기 HTTP 클라이언트)	비동기	• 애플리케이션에서 Kinesis API SubscribeToShard와 같이 HTTP/2 지원이 필요한 API를 호출하는 경우에 사용	다른 HTTP 클라이언트에 비해 시작 시간이 느림

HTTP 클라이언트	동기 또는 비동기	사용해야 하는 경우	제한 및 단점
AWS CRT 기반 비동기 HTTP 클라이언트[1]	비동기	• 애플리케이션이 AWS Lambda에서 실행 중인 경우 사용 • 짧은 지연 시간보다 높은 처리량을 선호하는 경우 사용 • SDK 클라이언트 비동기화를 선호하는 경우 사용	•`KinesisAsynClient` 및 `TranscribeStreamingAsyncClient` 와 같이 HTTP/2 지원이 필요한 서비스 클라이언트는 지원하지 않음 다음 Java 시스템 속성은 지원되지 않습니다. • javax.net.ssl.keyStore • javax.net.ssl.keyStorePassword • javax.net.ssl.trustStore • javax.net.ssl.trustStorePassword

[1]추가적인 이점 때문에 가능하면 AWS CRT 기반 HTTP 클라이언트를 사용하는 것이 좋습니다.

스마트 구성 기본값

AWS SDK for Java 2.x (버전 2.17.102 이상)는 스마트 구성 기본 기능을 제공합니다. 이 기능은 HTTP 클라이언트에 영향을 주지 않는 다른 속성과 함께 두 개의 HTTP 클라이언트 속성을 최적화합니다.

스마트 구성 기본값은 사용자가 제공한 기본값 모드 값을 기반으로 `connectTimeoutInMillis` 및 `tlsNegotiationTimeoutInMillis` 속성에 적절한 값을 설정합니다. 애플리케이션의 특성에 따라 기본 모드 값을 선택합니다.

스마트 구성 기본값 및 애플리케이션에 가장 적합한 기본값 모드 값을 선택하는 방법에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서를 참조하세요.

다음은 애플리케이션의 기본값 모드를 설정하는 네 가지 방법입니다.

Service client

서비스 클라이언트 빌더를 사용하여 서비스 클라이언트에서 직접 기본 모드를 구성하세요. 다음 예제에서는 `DynamoDbClient`에 대한 기본 출력 형식을 `auto`로 설정합니다.

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

System property

`aws.defaultsMode` 시스템 속성을 사용하여 기본 모드를 지정할 수 있습니다. Java에서 시스템 속성을 설정하는 경우 서비스 클라이언트를 초기화하기 전에 속성을 설정해야 합니다.

다음 예제에서는 Java에서 설정된 시스템 속성을 사용하여 기본 모드를 `auto`로 설정하는 방법을 보여줍니다.

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

다음 예제는 `java` 명령의 `-D` 옵션을 사용하여 기본값 모드를 `auto`로 설정하는 방법을 보여줍니다.

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

Environment variable

환경 변수 `AWS_DEFAULTS_MODE`의 값을 설정하여 애플리케이션의 기본 모드를 선택합니다.

다음 정보는 환경 변수를 `auto` 사용하여 기본값 모드의 값을 설정하기 위해 실행하는 명령을 보여줍니다.

운영 체제	환경 변수를 설정하는 방법
Linux, macOS 또는 Unix	`export AWS_DEFAULTS_MODE=auto`
Windows	`set AWS_DEFAULTS_MODE=auto`

AWS config file

다음 예제와 같이 공유 AWS `config` 파일에 `defaults_mode` 구성 속성을 추가할 수 있습니다.

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

시스템 속성, 환경 변수 또는 AWS 구성 파일을 사용하여 기본 모드를 전역으로 설정하는 경우 HTTP 클라이언트를 빌드할 때 설정을 재정의할 수 있습니다.

`httpClientBuilder()` 메서드로 HTTP 클라이언트를 빌드하면 빌드 중인 인스턴스에만 설정이 적용됩니다. 여기에 그 예제가 나와 있습니다. 이 예제의 Netty 기반 HTTP 클라이언트는 `connectTimeoutInMillis` 및 `tlsNegotiationTimeoutInMillis`에 대해 전역적으로 설정된 모든 기본 모드 값을 재정의합니다.

Apache 기반 HTTP 클라이언트 설정

AWS SDK for Java 2.x의 동기 서비스 클라이언트는 기본적으로 Apache 기반 HTTP 클라이언트, ApacheHttpClient를 사용합니다. SDK의 `ApacheHttpClient`는 Apache HTTPClient를 기반으로 합니다.

SDK는 로드 속도가 더 빠르지만 기능은 더 적은 UrlConnectionHttpClient를 제공합니다. `UrlConnectionHttpClient` 구성에 대한 내용은 the section called “URL 연결 기반 HTTP 클라이언트 구성”을 참조하세요.

`ApacheHttpClient`에서 사용할 수 있는 구성 옵션 전체를 보려면 ApacheHttpClient.Builder 및 ProxyConfiguration.Builder를 참조하세요.

`ApacheHttpClient` 액세스

대부분의 경우 명시적인 구성 없이 `ApacheHttpClient`를 사용합니다. 서비스 클라이언트를 선언하면 SDK가 표준 값으로 `ApacheHttpClient`를 구성합니다.

ApacheHttpClient를 명시적으로 구성하거나 여러 서비스 클라이언트와 함께 사용하려면 구성에 사용할 수 있도록 해야 합니다.

구성 불필요

Maven에서 서비스 클라이언트에 대한 종속성을 선언하면 SDK가 apache-client 아티팩트에 런타임 종속성을 추가합니다. 이렇게 하면 런타임에는 ApacheHttpClient 코드에서 클래스를 사용할 수 있지만 컴파일 타임에는 사용할 수 없습니다. Apache 기반 HTTP 클라이언트를 구성하지 않는 경우 종속성을 지정하지 않아도 됩니다.

Maven pom.xml 파일의 다음 XML 코드 조각에서 <artifactId>s3</artifactId>로 선언된 종속성은 자동적으로 Apache 기반 HTTP 클라이언트를 불러옵니다. 따로 종속성을 선언할 필요는 없습니다.

```
<dependencyManagement>
   <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>bom</artifactId>
            <version>2.27.21</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>
```

이러한 종속성을 사용하면 ApacheHttpClient 라이브러리가 런타임 클래스 경로에만 있으므로 HTTP 구성을 변경할 수 없습니다.

구성 필요

ApacheHttpClient를 구성하려면 컴파일 때 apache-client 라이브러리에 대한 종속성을 추가해야 합니다.

ApacheHttpClient를 구성하려면 다음 Maven pom.xml 파일 예제를 참조하세요.

```
    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>software.amazon.awssdk</groupId>
                <artifactId>bom</artifactId>
                <version>2.27.21</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>
```

ApacheHttpClient 사용 및 구성

서비스 클라이언트 구축과 함께 ApacheHttpClient의 인스턴스를 구성하거나 여러 서비스 클라이언트에서 공유하도록 단일 인스턴스를 구성할 수 있습니다.

어느 방법을 사용하든 ApacheHttpClient.Builder를 사용하여 Apache 기반 HTTP 클라이언트의 속성을 구성합니다.

모범 사례: 서비스 클라이언트 전용 **ApacheHttpClient** 인스턴스 지정

ApacheHttpClient의 인스턴스를 구성해야 하는 경우 전용 ApacheHttpClient 인스턴스를 구축하는 것이 좋습니다. 서비스 클라이언트 빌더의 httpClientBuilder 메서드를 사용하면 됩니다. 이렇게 하면 SDK에서 HTTP 클라이언트의 수명 주기를 관리하므로 더 이상 필요하지 않을 때 ApacheHttpClient 인스턴스를 종료하지 않을 경우 잠재적인 메모리 누수를 방지할 수 있습니다.

다음 예제에서는 S3Client 인스턴스를 생성하고 maxConnections 및 connectionTimeout 값과 함께 ApacheHttpClient의 내장 인스턴스를 구성합니다. HTTP 인스턴스는 S3Client.Builder의 httpClientBuilder 메서드를 사용하여 생성됩니다.

가져옵니다.

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

코드

```
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.
```

대안 접근 방식: **ApacheHttpClient** 인스턴스 공유

애플리케이션의 리소스 및 메모리 사용량을 낮추려면 ApacheHttpClient를 구성하고 여러 서비스 클라이언트에서 공유할 수 있습니다. HTTP 연결 풀이 공유되므로 리소스 사용량이 줄어듭니다.

> **Note**
>
> ApacheHttpClient 인스턴스를 공유한 경우 폐기할 준비가 되면 인스턴스를 닫아야 합니다. SDK는 서비스 클라이언트가 닫힐 때 인스턴스를 닫지 않습니다.

다음 예제는 두 서비스 클라이언트에서 사용하는 Apache 기반 HTTP 클라이언트를 구성하는 예제입니다. 구성된 ApacheHttpClient 인스턴스는 각 빌더의 httpClient 메서드로 전달됩니다. 서비스 클라이언트와 HTTP 클라이언트가 더 이상 필요하지 않으면 코드가 명시적으로 닫습니다. 코드는 HTTP 클라이언트를 마지막으로 닫습니다.

가져옵니다.

```
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;
```

코드

```
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.
```

프록시 구성 예제

다음 코드 조각은 Apache HTTP 클라이언트용 프록시 구성 빌더를 사용합니다.

```
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();
```

프록시 구성에 해당하는 Java 시스템 속성은 다음 명령줄 코드 조각에 나와 있습니다.

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

환경 변수를 사용하는 것과 동등한 설정은 다음과 같습니다.

```
// 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**
>
> Apache HTTP 클라이언트는 현재 HTTPS 프록시 시스템 속성이나 HTTPS_PROXY 환경 변수를 지원하지 않습니다.

URL 연결 기반 HTTP 클라이언트 구성

AWS SDK for Java 2.x는 기본값 `ApacheHttpClient`에 비해 더 가벼운 `UrlConnectionHttpClient` HTTP 클라이언트를 제공합니다. `UrlConnectionHttpClient`는 Java의 `URLConnection`를 기반으로 합니다.

`UrlConnectionHttpClient`는 Apache 기반 HTTP 클라이언트보다 로드 속도가 빠르지만 기능이 더 적습니다. 로드 속도가 더 빠르기 때문에 Java AWS Lambda 함수에 적합한 솔루션입니다.

`UrlConnectionHttpClient`에는 액세스할 수 있는 여러 구성 가능한 옵션이 있습니다.

> **Note**
>
> `UrlConnectionHttpClient`에서는 HTTP PATCH 메서드를 지원하지 않습니다.

일부 AWS API 작업에는 PATCH 요청이 필요합니다. 이러한 작업 이름은 일반적으로 `Update*`로 시작합니다. 다음은 몇 가지 예제입니다.

- AWS Security Hub CSPM API에서의 몇 가지 `Update*` 작업과 BatchUpdateFindings 작업
- 모든 Amazon API Gateway API `Update*` 작업

`UrlConnectionHttpClient`를 사용할 수 있는 경우 먼저 사용 중인 AWS 서비스의 API 참조를 참조하세요. 필요한 작업이 PATCH 작업을 사용하는지 확인하세요.

`UrlConnectionHttpClient` 액세스

`UrlConnectionHttpClient`를 구성하고 사용하려면 `pom.xml` 파일의 `url-connection-client` Maven 아티팩트에 대한 종속성을 선언해야 합니다.

`ApacheHttpClient`와 달리 `UrlConnectionHttpClient`는 프로젝트에 자동으로 추가되지 않으므로 사용자는 이를 구체적으로 선언해야 합니다.

`pom.xml` 파일의 다음 예제는 HTTP 클라이언트를 사용하고 구성하는 데 필요한 종속성을 보여줍니다.

```
<dependencyManagement>
   <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>bom</artifactId>
            <version>2.27.21</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>
```

`UrlConnectionHttpClient` 사용 및 구성

서비스 클라이언트 구축과 함께 `UrlConnectionHttpClient`의 인스턴스를 구성하거나 여러 서비스 클라이언트에서 공유하도록 단일 인스턴스를 구성할 수 있습니다.

어느 방법을 사용하든 URLConnectionHttpClient.Builder를 사용하여 URLConnection 기반 HTTP 클라이언트의 속성을 구성합니다.

모범 사례: 서비스 클라이언트 전용 **`UrlConnectionHttpClient`** 인스턴스 지정

`UrlConnectionHttpClient`의 인스턴스를 구성해야 하는 경우 전용 `UrlConnectionHttpClient` 인스턴스를 구축하는 것이 좋습니다. 서비스 클라이언트 빌더의 `httpClientBuilder` 메서드를 사용하면 됩니다. 이렇게 하면 SDK에서 HTTP 클라이언트의 수명 주기를 관리하므로 더 이상 필요하지 않을 때 `UrlConnectionHttpClient` 인스턴스를 종료하지 않을 경우 잠재적인 메모리 누수를 방지할 수 있습니다.

다음 예제에서는 `S3Client` 인스턴스를 생성하고 `socketTimeout` 및 `proxyConfiguration` 값과 함께 `UrlConnectionHttpClient`의 내장 인스턴스를 구성합니다. 이 `proxyConfiguration` 메서드는 `Consumer<ProxyConfiguration.Builder>` 유형의 Java 람다 표현식을 사용합니다.

가져옵니다.

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

코드

```
// 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.
```

대안 접근 방식: **`UrlConnectionHttpClient`** 인스턴스 공유

애플리케이션의 리소스 및 메모리 사용량을 낮추려면 `UrlConnectionHttpClient`를 구성하고 여러 서비스 클라이언트에서 공유할 수 있습니다. HTTP 연결 풀이 공유되므로 리소스 사용량이 줄어듭니다.

> **Note**
>
> `UrlConnectionHttpClient` 인스턴스를 공유한 경우 폐기할 준비가 되면 인스턴스를 닫아야 합니다. SDK는 서비스 클라이언트가 닫힐 때 인스턴스를 닫지 않습니다.

다음 예제는 두 서비스 클라이언트가 사용하는 URLConnection 기반 HTTP 클라이언트를 구성합니다. 구성된 `UrlConnectionHttpClient` 인스턴스는 각 빌더의 `httpClient` 메서드로 전달됩니다. 서비스 클라이언트와 HTTP 클라이언트가 더 이상 필요하지 않으면 코드가 명시적으로 닫습니다. 코드는 HTTP 클라이언트를 마지막으로 닫습니다.

가져옵니다.

```
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;
```

코드

```
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();
```

URLConnectionHttpClient를 **ApacheHttpClient**와 함께 사용

애플리케이션에서 UrlConnectionHttpClient를 사용하는 경우 서비스 클라이언트 빌더의 httpClientBuilder 메서드를 사용하여 각 서비스 클라이언트에 URLConnectionHttpClient 인스턴스 또는 ApacheHttpClient 인스턴스를 제공해야 합니다.

프로그램에서 여러 서비스 클라이언트를 사용하고 다음 두 가지 조건을 모두 만족할 때 예외가 발생합니다.

- 한 서비스 클라이언트가 UrlConnectionHttpClient 인스턴스를 사용하도록 구성되어 있습니다.
- 다른 서비스 클라이언트는 httpClient() 또는 httpClientBuilder() 메서드로 명시적으로 빌드하지 않고 기본값 ApacheHttpClient을 사용합니다.

예외는 클래스 경로에서 여러 HTTP 구현이 발견되었음을 나타냅니다.

다음 예제 코드 조각은 예외로 이어집니다.

```
// 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();
```

S3Client를 ApacheHttpClient로 명시적으로 구성하여 예외를 방지하세요.

```
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**
>
> ApacheHttpClient를 명시적으로 만들려면 Maven 프로젝트 파일의 apache-client아티팩트에 대한 종속성을 추가해야 합니다.

프록시 구성 예제

다음 코드 조각은 URL 연결 HTTP 클라이언트용 프록시 구성 빌더를 사용합니다.

```
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();
```

프록시 구성에 해당하는 Java 시스템 속성은 다음 명령줄 코드 조각에 나와 있습니다.

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

환경 변수를 사용하는 것과 동등한 설정은 다음과 같습니다.

```
// 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**
>
> URL 연결 기반 HTTP 클라이언트는 현재 HTTPS 프록시 시스템 속성이나 HTTPS_PROXY 환경 변수를 지원하지 않습니다.

Netty 기반 HTTP 클라이언트를 구성

AWS SDK for Java 2.x에서 비동기 작업을 위한 기본 HTTP 클라이언트는 Netty 기반 NettyNioAsyncHttpClient입니다. Netty 기반 클라이언트는 Netty 프로젝트의 비동기 이벤트 기반 네트워크 프레임워크를 기반으로 합니다.

대체 HTTP 클라이언트로서 새 AWSCRT 기반 HTTP 클라이언트를 사용할 수 있습니다. 이 항목에서는 `NettyNioAsyncHttpClient`를 구성하는 방법을 보여 줍니다.

`NettyNioAsyncHttpClient` 액세스

대부분의 경우 비동기 프로그램에서는 명시적인 구성 없이 `NettyNioAsyncHttpClient`를 사용합니다. 비동기 서비스 클라이언트를 선언하면 SDK가 표준 값으로 `NettyNioAsyncHttpClient`를 구성합니다.

`NettyNioAsyncHttpClient`를 명시적으로 구성하거나 여러 서비스 클라이언트와 함께 사용하려면 구성에 사용할 수 있도록 해야 합니다.

구성 불필요

Maven에서 서비스 클라이언트에 대한 종속성을 선언하면 SDK가 `netty-nio-client` 아티팩트에 런타임 종속성을 추가합니다. 이렇게 하면 런타임에는 `NettyNioAsyncHttpClient` 코드에서 클래스를 사용할 수 있지만 컴파일 타임에는 사용할 수 없습니다. Netty 기반 HTTP 클라이언트를 구성하지 않는 경우 종속성을 지정하지 않아도 됩니다.

Maven `pom.xml` 파일의 다음 XML 코드 조각에서 `<artifactId>dynamodb-enhanced</artifactId>`로 선언된 종속성은 전이적으로 Netty 기반 HTTP 클라이언트를 불러옵니다. 따로 종속성을 선언할 필요는 없습니다.

```
<dependencyManagement>
   <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>bom</artifactId>
            <version>2.27.21</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
<dependencies>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>dynamodb-enhanced</artifactId>
    </dependency>
</dependencies>
```

이러한 종속성을 사용하면 `NettyNioAsyncHttpClient` 라이브러리가 런타임 클래스 경로에만 있으므로 HTTP 구성을 변경할 수 없습니다.

구성 필요

NettyNioAsyncHttpClient를 구성하려면 컴파일 때 netty-nio-client 아티팩트에 대한 종속성을 추가해야 합니다.

NettyNioAsyncHttpClient를 구성하려면 다음 Maven pom.xml 파일 예제를 참조하세요.

```
    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>software.amazon.awssdk</groupId>
                <artifactId>bom</artifactId>
                <version>2.27.21</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>
```

NettyNioAsyncHttpClient 사용 및 구성

서비스 클라이언트 구축과 함께 NettyNioAsyncHttpClient의 인스턴스를 구성하거나 여러 서비스 클라이언트에서 공유하도록 단일 인스턴스를 구성할 수 있습니다.

어느 방법을 사용하든 NettyNIOAsynchttpClient.Builder를 사용하여 Netty 기반 HTTP 클라이언트 인스턴스의 속성을 구성합니다.

모범 사례: 서비스 클라이언트 전용 **NettyNioAsyncHttpClient** 인스턴스 지정

NettyNioAsyncHttpClient의 인스턴스를 구성해야 하는 경우 전용 NettyNioAsyncHttpClient 인스턴스를 구축하는 것이 좋습니다. 서비스 클라이언트 빌더의 httpClientBuilder 메서드를 사용하면 됩니다. 이렇게 하면 SDK에서 HTTP 클라이언트의 수명 주기를 관리하므로 더 이상 필요하지 않을 때 NettyNioAsyncHttpClient 인스턴스를 종료하지 않을 경우 잠재적인 메모리 누수를 방지할 수 있습니다.

다음 예시에서는 DynamoDbEnhancedAsyncClient 인스턴스에서 사용되는 DynamoDbAsyncClient 인스턴스를 만듭니다. DynamoDbAsyncClient 인스턴스에는 connectionTimeout 및 maxConcurrency 값이 있는 NettyNioAsyncHttpClient 인스턴스가 포함됩니다. HTTP 인스턴스는 DynamoDbAsyncClient.Builder의 httpClientBuilder 메서드를 사용하여 생성됩니다.

가져옵니다.

```
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;
```

코드

```
// 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();
```

대안 접근 방식: **NettyNioAsyncHttpClient** 인스턴스 공유

애플리케이션의 리소스 및 메모리 사용량을 낮추려면 NettyNioAsyncHttpClient를 구성하고 여러 서비스 클라이언트에서 공유할 수 있습니다. HTTP 연결 풀이 공유되므로 리소스 사용량이 줄어듭니다.

> **Note**
>
> NettyNioAsyncHttpClient 인스턴스를 공유한 경우 폐기할 준비가 되면 인스턴스를 닫아야 합니다. SDK는 서비스 클라이언트가 닫힐 때 인스턴스를 닫지 않습니다.

다음 예제는 두 서비스 클라이언트가 사용하는 Netty 기반 HTTP 클라이언트를 구성합니다. 구성된 NettyNioAsyncHttpClient 인스턴스는 각 빌더의 httpClient 메서드로 전달됩니다. 서비스 클라이언트와 HTTP 클라이언트가 더 이상 필요하지 않으면 코드가 명시적으로 닫습니다. 코드는 HTTP 클라이언트를 마지막으로 닫습니다.

가져옵니다.

```
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;
```

코드

```
// 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.
```

ALPN 프로토콜 협상 구성

ALPN(Application-Layer Protocol Negotiation)은 애플리케이션 계층에서 추가 왕복을 방지하고 더 나은 성능을 제공하는 방식으로 보안 연결을 통해 수행해야 하는 프로토콜을 협상할 수 있는 TLS 확장 프로그램입니다.

Netty 기반 HTTP 클라이언트가 ALPN을 사용하도록 설정하려면 다음 코드 조각과 같이 빌더 메서드를 호출합니다.

```
import software.amazon.awssdk.http.Protocol;
import software.amazon.awssdk.http.ProtocolNegotiation;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
```

```
import
 software.amazon.awssdk.services.transcribestreaming.TranscribeStreamingAsyncClient;


// Configure the Netty-based HTTP client to use the ALPN protocol.
SdkAsyncHttpClient nettyClient = NettyNioAsyncHttpClient.builder()
                                                        .protocol(Protocol.HTTP2)

 .protocolNegotiation(ProtocolNegotiation.ALPN)
                                                        .build();
// Use the Netty-based HTTP client with a service client.
TranscribeStreamingAsyncClient transcribeClient =
 TranscribeStreamingAsyncClient.builder()

 .httpClient(nettyClient)

 .build();
```

ALPN 프로토콜 협상은 현재 이전 코드 조각과 같이 HTTP/2 프로토콜에서만 작동합니다.

프록시 구성 예제

다음 코드 조각은 Netty HTTP 클라이언트용 프록시 구성 빌더를 사용합니다.

```
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();
```

프록시 구성에 해당하는 Java 시스템 속성은 다음 명령줄 코드 조각에 나와 있습니다.

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

> **Important**
>
> HTTPS 프록시 시스템 속성을 사용하려면 코드에서 `scheme` 속성을 `https`로 설정해야 합니다. 스키마 속성이 코드에 설정되지 않은 경우 스키마는 HTTP로 기본 설정되며 SDK는 `http.*` 시스템 속성만 찾습니다.

환경 변수를 사용하는 것과 동등한 설정은 다음과 같습니다.

```
// 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
```

AWS CRT 기반 HTTP 클라이언트 구성

AWS CRT 기반 HTTP 클라이언트에는 동기식 AwsCrtHttpClient와 비동기식 AwsCrtAsyncHttpClient가 있습니다. AWS CRT 기반 HTTP 클라이언트는 다음과 같은 HTTP 클라이언트 이점을 제공합니다.

- 더 빠른 SDK 시작 시간
- 더 작은 메모리 공간
- 대기 시간 단축
- 연결 상태 관리
- DNS 로드 밸런싱

SDK의 AWS CRT 기반 구성 요소

이 항목에 설명된 AWS CRT 기반 HTTP 클라이언트와 AWS CRT 기반 S3 클라이언트는 SDK의 서로 다른 구성 요소입니다.

동기식 및 비동기식 AWS CRT 기반 HTTP 클라이언트는 SDK HTTP 클라이언트 인터페이스를 구현한 것으로, 일반적인 HTTP 통신에 사용됩니다. SDK의 다른 동기 또는 비동기 HTTP 클라이언트에 대한 대안으로 추가적인 이점이 있습니다.

AWS CRT 기반 S3 클라이언트는 S3AsyncClient 인터페이스를 구현한 것으로, Amazon S3 서비스를 사용하는 데 사용됩니다. 이는 `S3AsyncClient` 인터페이스의 Java 기반 구현의 대안이며 여러 가지 이점을 제공합니다.

두 구성 요소 모두 AWS 공용 런타임의 라이브러리를 사용하지만 AWS CRT 기반 HTTP 클라이언트는 aws-c-s3 라이브러리를 사용하지 않으며 S3 멀티파트 업로드 API 기능을 지원하지 않습니다. 반면 AWS CRT 기반 S3 클라이언트는 S3 멀티파트 업로드 API 기능을 지원하도록 특별히 구축되었습니다.

AWS CRT 기반 HTTP 클라이언트에 액세스

AWS CRT 기반 HTTP 클라이언트를 사용하려면 먼저 프로젝트의 종속성에 최소 버전 2.22.0의 `aws-crt-client` 아티팩트를 추가하세요.

다음 옵션 중 하나를 사용하여 Maven `pom.xml` 파일을 설정합니다.

> **Note**
>
> 예를 들어 애플리케이션이 AWS Lambda 함수에서 실행되는 경우 런타임 종속성의 크기를 더 작게 유지해야 하면 플랫폼별 jar 옵션을 사용하도록 선택할 수 있습니다.

Uber-jar option

기본적으로는 Linux, Windows 및 macOS 등 여러 플랫폼의 바이너리가 포함된 AWS CRT 아티팩트의 uber-jar를 `aws-crt-client`에서 사용합니다.

```
<project>
   <properties>
     <aws.sdk.java.version>2.29.10*</aws.sdk.java.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>
```

*빨간색으로 표시된 버전을 사용하려는 Java SDK 버전으로 바꿉니다. Maven Central에서 최신 정보를 찾습니다.

Platform-specific jar option

Java 런타임을 AWS CRT 라이브러리의 플랫폼별 버전으로 제한하려면 Uber-jar 옵션을 다음과 같이 변경합니다.

- SDK의 `aws-crt-client` 아티팩트에 `exclusions` 요소를 추가합니다. 이 제외 작업은 SDK가 AWS CRT uber-jar를 전이적으로 사용하는 것을 방지합니다.
- 필요한 특정 AWS CRT 플랫폼 버전에 대한 종속성 요소를 추가합니다. 올바른 버전을 결정하는 방법은 아래 AWS CRT 아티팩트 버전을 결정하는 단계를 참조하세요.

```
<project>
   <properties>
     <aws.sdk.java.version>2.29.10[1]</aws.sdk.java.version>
  </properties>
   <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>software.amazon.awssdk</groupId>
                <artifactId>bom</artifactId>
                <version>${aws.sdk.java.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
```

```
            <artifactId>aws-crt-client</artifactId>
            <exclusions>
                <exclusion>
                    <groupId>software.amazon.awssdk.crt</groupId>
                    <artifactId>aws-crt</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
        <dependency>
            <groupId>software.amazon.awssdk.crt</groupId>
            <artifactId>aws-crt</artifactId>
            <version>0.31.3[2]</version>
            <classifier>linux-x86_64[3]</classifier>
        </dependency>
    </dependencies>
```

[1]빨간색으로 표시된 버전을 사용하려는 Java SDK 버전으로 바꿉니다. Maven Central에서 최신 정보를 찾습니다.

[2]Uber-jar 옵션에서 제공하는 `software.amazon.awssdk.crt:aws-crt`의 버전을 바꿉니다. AWS CRT 아티팩트 버전을 확인하는 단계는 다음을 참조하세요.

[3]`classifier` 값을 플랫폼의 값으로 바꿉니다. 사용 가능한 값 목록은 AWS CRT for Java GitHub 페이지를 참조하세요.

AWS CRT 아티팩트 버전을 확인하는 단계

다음 단계를 사용하여 사용 중인 SDK for Java 버전과 호환되는 AWS CRT 아티팩트 버전을 확인합니다.

1. Uber-jar 옵션에 표시된 대로 `pom.xml` 파일을 설정합니다. 이 설정을 통해 SDK가 기본적으로 제공하는 `software.amazon.awssdk.crt:aws-crt`의 버전을 확인할 수 있습니다.
2. 프로젝트의 루트(`pom.xml` 파일과 동일한 디렉터리)에서 다음 Maven 명령을 실행합니다.

   ```
   mvn dependency:tree -Dincludes=software.amazon.awssdk.crt:aws-crt
   ```

   Maven은 다른 작업을 수행할 수 있지만, 결국 SDK가 전이적으로 사용하는 `software.amazon.awssdk.crt:aws-crt` 종속성의 콘솔 출력이 표시되어야 합니다. 다음 코드 조각은 `2.29.10`의 SDK 버전을 기반으로 한 샘플 출력을 보여줍니다.

   ```
   [INFO] org.example:yourProject:jar:1.0-SNAPSHOT
   ```

```
[INFO] \- software.amazon.awssdk:aws-crt-client:jar:2.29.10:compile
[INFO]    \- software.amazon.awssdk.crt:aws-crt:jar:0.31.3:compile
```

3. 콘솔에 `software.amazon.awssdk.crt:aws-crt` 아티팩트에 대해 표시되는 버전을 사용합니다. 이 경우 `pom.xml` 파일에 `0.31.3`을 추가합니다.

AWS CRT 기반 HTTP 클라이언트 사용 및 구성

서비스 클라이언트를 빌드할 때 AWS CRT 기반 HTTP 클라이언트를 구성하거나 단일 인스턴스를 구성하여 여러 서비스 클라이언트에서 공유할 수 있습니다.

어느 방법을 사용하든 빌더를 사용하여 AWS CRT 기반 HTTP 클라이언트 인스턴스의 속성을 구성합니다.

모범 사례: 서비스 클라이언트 전용 인스턴스 지정

AWS CRT 기반 HTTP 클라이언트의 인스턴스를 구성해야 하는 경우 서비스 클라이언트와 함께 빌드하여 인스턴스를 전용으로 사용하는 것이 좋습니다. 서비스 클라이언트 빌더의 `httpClientBuilder` 메서드를 사용하면 됩니다. 이렇게 하면 HTTP 클라이언트의 수명 주기가 SDK에 의해 관리되므로 더 이상 필요하지 않을 때 AWS CRT 기반 HTTP 클라이언트 인스턴스를 종료하지 않을 경우 메모리 누수 가능성을 방지하는 데 도움이 됩니다.

다음 예제에서는 S3 서비스 클라이언트를 생성하고 `connectionTimeout` 및 `maxConcurrency` 값을 사용하여 AWS CRT 기반 HTTP 클라이언트를 구성합니다.

Synchronous client

가져옵니다.

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

코드

```
// 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

가져옵니다.

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

코드

```
// 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();
```

대안 접근 방식: 인스턴스 공유

애플리케이션의 리소스 및 메모리 사용량을 낮추려면 AWS CRT 기반 HTTP 클라이언트를 구성하고 여러 서비스 클라이언트에서 공유할 수 있습니다. HTTP 연결 풀이 공유되므로 리소스 사용량이 줄어듭니다.

> **Note**
>
> AWS CRT 기반 HTTP 클라이언트 인스턴스를 공유한 경우 폐기할 준비가 되면 인스턴스를 닫아야 합니다. SDK는 서비스 클라이언트가 닫힐 때 인스턴스를 닫지 않습니다.

다음 예제는 connectionTimeout 및 maxConcurrency 값을 사용하여 AWS CRT 기반 HTTP 클라이언트 인스턴스를 구성합니다. 구성된 인스턴스는 각 서비스 클라이언트 빌더의 httpClient 메서드로 전달됩니다. 서비스 클라이언트와 HTTP 클라이언트가 더 이상 필요하지 않으면 명시적으로 닫힙니다. HTTP 클라이언트가 마지막으로 닫힙니다.

Synchronous client

가져옵니다.

```
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;
```

코드

```
// 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

가져옵니다.

```
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;
```

코드

```
// 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.
```

AWS CRT 기반 HTTP 클라이언트를 기본값으로 설정

SDK가 서비스 클라이언트에 대한 기본 HTTP 클라이언트로 AWS CRT 기반 HTTP 클라이언트를 사용하도록 Maven 빌드 파일을 설정할 수 있습니다.

이렇게 하려면 각 서비스 클라이언트 아티팩트에 기본 HTTP 클라이언트 종속성이 있는 `exclusions` 요소를 추가하면 됩니다.

다음 `pom.xml` 예제에서 SDK는 S3 서비스에 대해 AWS CRT 기반 HTTP 클라이언트를 사용합니다. 코드의 서비스 클라이언트가 `S3AsyncClient`인 경우, SDK는 `AwsCrtAsyncHttpClient`를 사용합니다. 서비스 클라이언트가 S3Client인 경우 SDK는 `AwsCrtHttpClient`를 사용합니다. 이 설정에서는 기본 Netty 기반 비동기 HTTP 클라이언트와 기본 Apache 기반 동기 HTTP를 사용할 수 없습니다.

```
<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>
```

최신 *VERSION* 값을 보려면 Maven 중앙 리포지토리를 방문하세요.

> **Note**
>
> 여러 서비스 클라이언트가 `pom.xml` 파일에 선언되어 있는 경우 모두 `exclusions` XML 요소가 필요합니다.

Java 시스템 속성 사용

AWS CRT 기반 HTTP 클라이언트를 애플리케이션의 기본 HTTP로 사용하려면 Java 시스템 속성 `software.amazon.awssdk.http.async.service.impl`을 `software.amazon.awssdk.http.crt.AwsCrtSdkHttpService` 값으로 설정하면 됩니다.

애플리케이션 시작 중에 설정하려면 다음과 유사한 명령을 실행하세요.

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

다음 코드 조각을 사용하여 애플리케이션 코드에서 시스템 속성을 설정합니다.

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

> **Note**
>
> 시스템 속성을 사용하여 AWS CRT 기반 HTTP 클라이언트 사용을 구성할 때는 `poml.xml` 파일의 `aws-crt-client` 아티팩트에 종속성을 추가해야 합니다.

AWS CRT 기반 HTTP 클라이언트의 고급 구성

연결 상태 구성 및 최대 유휴 시간 등 AWS CRT 기반 HTTP 클라이언트의 다양한 구성 설정을 사용할 수 있습니다. `AwsCrtAsyncHttpClient`에서 사용할 수 있는 구성 옵션을 검토할 수 있습니다. `AwsCrtHttpClient`에 사용할 수 있는 구성 옵션을 검토할 수 있습니다.

연결 상태 구성

HTTP 클라이언트 빌더의 `connectionHealthConfiguration` 방법을 사용하여 AWS CRT 기반 HTTP 클라이언트의 연결 상태 구성을 구성할 수 있습니다.

다음 예는 연결 상태 구성과 연결 최대 유휴 시간으로 구성된 AWS CRT 기반 HTTP 클라이언트 인스턴스를 사용하는 S3 서비스 클라이언트를 만드는 예제입니다.

Synchronous client

가져옵니다.

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

코드

```
// 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

가져옵니다.

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

코드

```
// 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 지원

HTTP/2 프로토콜은 아직 AWS CRT 기반 HTTP 클라이언트에서 지원되지 않지만 향후 릴리즈에서 지원될 예정입니다.

그동안 KinesisAsyncClient 또는 TranscribeStreamingAsyncClient와 같이 HTTP/2 지원이 필요한 서비스 클라이언트를 사용하는 경우 NettyNioAsyncHttpClient를 대신 사용하는 것이 좋습니다.

프록시 구성 예제

다음 코드 조각은 코드에서 프록시 설정을 구성하는 데 사용하는 `ProxyConfiguration.Builder`의 사용법을 보여줍니다.

Synchronous client

가져옵니다.

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

코드

```
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

가져옵니다.

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

코드

```
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();
```

프록시 구성에 해당하는 Java 시스템 속성은 다음 명령줄 코드 조각에 나와 있습니다.

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

> **Important**
>
> HTTPS 프록시 시스템 속성을 사용하려면 코드에서 `scheme` 속성을 `https`로 설정해야 합니다. 스키마 속성이 코드에 설정되지 않은 경우 스키마는 HTTP로 기본 설정되며 SDK는 `http.*` 시스템 속성만 찾습니다.

환경 변수를 사용하는 것과 동등한 설정은 다음과 같습니다.

```
// 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
```

HTTP 프록시 구성

코드를 사용하거나, Java 시스템 속성을 설정하거나, 환경 변수를 설정하여 HTTP 프록시를 구성할 수 있습니다.

코드로 구성

서비스 클라이언트를 빌드할 때 클라이언트별 `ProxyConfiguration` 빌더를 사용하여 코드로 프록시를 구성합니다. 다음 예제는 Amazon S3 서비스 클라이언트에서 사용하는 Apache 기반 HTTP 클라이언트의 프록시 구성 예제를 보여줍니다.

```
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();
```

이 항목의 각 HTTP 클라이언트 단원에서는 프록시 구성 예제를 보여줍니다.

- Apache 기반 HTTP 클라이언트
- URLConnection 기반 HTTP 클라이언트
- Netty 기반 HTTP 클라이언트
- AWS CRT 기반 HTTP 클라이언트

외부 설정을 사용하여 HTTP 프록시 구성

코드에서 `ProxyConfiguration` 빌더를 명시적으로 사용하지 않더라도 SDK는 외부 설정을 찾아 기본 프록시 구성을 구성합니다.

기본적으로 SDK는 먼저 JVM 시스템 속성을 검색합니다. 속성이 하나라도 발견되면 SDK는 값과 기타 시스템 속성 값을 사용합니다. 사용 가능한 시스템 속성이 없는 경우 SDK는 프록시 환경 변수를 찾습니다.

SDK는 다음 Java 시스템 속성 및 환경 변수를 사용할 수 있습니다.

Java 시스템 속성

시스템 속성	설명	HTTP 클라이언트 지원
http.proxyHost	HTTPS 프록시 서버의 호스트 이름	모두
http.proxyPort	HTTPS 프록시 서버의 포트 번호	모두
http.proxyUser	HTTPS 프록시 인증을 위한 사용자 이름	모두
http.proxyPassword	HTTPS 프록시 인증을 위한 비밀번호	모두
http.nonProxyHosts	프록시를 우회하여 직접 연결해야 하는 호스트 목록. 이 목록은 HTTPS를 사용하는 경우에도 유효합니다.	모두
https.ProxyHost	HTTPS 프록시 서버의 호스트 이름	Netty, CRT

시스템 속성	설명	HTTP 클라이언트 지원
https.ProxyPort	HTTPS 프록시 서버의 포트 번호	Netty, CRT
https.proxyUser	HTTPS 프록시 인증을 위한 사용자 이름	Netty, CRT
https.proxyPassword	HTTPS 프록시 인증을 위한 비밀번호	Netty, CRT

환경 변수

환경 변수	설명	HTTP 클라이언트 지원
HTTP_PROXY[1]	HTTP의 스키마가 있는 유효한 URL	모두
HTTPS_PROXY[1]	HTTPS의 스키마가 있는 유효한 URL	Netty, CRT
NO_PROXY[2]	프록시를 우회하여 직접 연결해야 하는 호스트 목록. 목록은 HTTP와 HTTPS 모두에 유효합니다.	모두

키 및 각주 보기

모두: SDK에서 제공하는 모든 HTTP 클라이언트 - `UrlConnectionHttpClient`, `ApacheHttpClient`, `NettyNioAsyncHttpClient`, `AwsCrtAsyncHttpClient`

Netty: Netty 기반 HTTP 클라이언트(`NettyNioAsyncHttpClient`)

CRT: AWS CRT 기반 HTTP 클라이언트(`AwsCrtHttpClient` 및 `AwsCrtAsyncHttpClient`)

[1]쿼리된 환경 변수는 `HTTP_PROXY` 또는 `HTTPS_PROXY`에 관계없이 클라이언트의 `ProxyConfiguration`에 있는 스키마 설정에 따라 달라집니다. 기본 스키마는 HTTP입니다. 다음 코드 조각은 스키마를 환경 변수 해결에 사용되는 HTTPS로 변경하는 방법을 보여줍니다.

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

[2]NO_PROXY 환경 변수는 호스트 이름 간에 '|' 및 ',' 구분 기호의 조합을 지원합니다. 호스트 이름에는 '*' 와일드카드가 포함될 수 있습니다.

설정 조합 사용

코드, 시스템 속성 및 환경 변수에서 HTTP 프록시 설정 조합을 사용할 수 있습니다.

Example- 시스템 속성과 코드로 제공되는 구성

```
// 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();
```

SDK는 다음 프록시 설정을 해결합니다.

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

Example- 시스템 속성과 환경 변수 모두 사용

각 HTTP 클라이언트의 `ProxyConfiguration` 빌더는 `useSystemPropertyValues` 및 `useEnvironmentVariablesValues`라는 설정을 제공합니다. 기본적으로 두 설정은 모두 `true`입니다. `true`로 설정하면 SDK는 `ProxyConfiguration` 빌더를 통해 제공되지 않은 옵션에 대해 시스템 속성의 값 또는 환경 변수를 자동으로 사용합니다.

> **Important**
>
> 이 시스템 속성은 환경 변수보다 우선합니다. HTTP 프록시 시스템 속성이 발견되면 SDK는 시스템 속성에서 모든 값을 검색하고 환경 변수에서는 검색하지 않습니다. 시스템 속성보다 환경 변수를 우선하도록 순위를 지정하려면 `useSystemPropertyValues`를 `false`로 설정합니다.

이 예제에서는 런타임에서 다음 설정을 사용할 수 있습니다.

```
// 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"
```

서비스 클라이언트는 다음 문 중 하나를 사용해 만듭니다. 모든 문은 프록시 설정을 명시적으로 설정하지 않습니다.

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

SDK는 다음 프록시 설정을 확인합니다.

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

서비스 클라이언트에는 기본 프록시 설정이 있으므로 SDK는 시스템 속성을 검색한 다음 환경 변수를 검색합니다. 시스템 속성 설정이 환경 변수보다 우선하므로 SDK는 시스템 속성만 사용합니다.

다음 코드와 같이 시스템 속성의 사용이 `false`로 변경되면 SDK는 환경 변수만 해결합니다.

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

HTTP를 사용하여 해결된 프록시 설정은 다음과 같습니다.

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

Apache 5.x 기반 HTTP 클라이언트 구성

Apache5HttpClient 액세스

를 사용하려면에 대한 종속성을 추가apache5-client하고 서비스 클라이언트`Apache5HttpClient`에 대해를 명시적으로 구성`Apache5HttpClient`해야 합니다.

```
<dependencyManagement>
   <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>bom</artifactId>
            <version>2.41.0*</version>
```

```
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>

<dependencies>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>s3</artifactId>
    </dependency>

    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>apache5-client</artifactId>
    </dependency>
</dependencies>
```

*빨간색으로 표시된 버전을 사용하려는 Java SDK 버전으로 바꿉니다. Maven Central에서 최신 정보를 찾습니다.

`Apache5HttpClient` 사용 및 구성

서비스 클라이언트 구축과 함께 `Apache5HttpClient`의 인스턴스를 구성하거나 여러 서비스 클라이언트에서 공유하도록 단일 인스턴스를 구성할 수 있습니다.

어느 접근 방식이든 Apache5HttpClient.Builder를 사용하여 Apache 5 기반 HTTP 클라이언트의 속성을 구성합니다.

모범 사례: Apache5HttpClient 인스턴스를 서비스 클라이언트에 전용

`Apache5HttpClient`의 인스턴스를 구성해야 하는 경우 전용 `Apache5HttpClient` 인스턴스를 구축하는 것이 좋습니다. 서비스 클라이언트 빌더의 httpClientBuilder 메서드를 사용하여이 작업을 수행할 수 있습니다. 이렇게 하면 SDK에서 HTTP 클라이언트의 수명 주기를 관리하므로 더 이상 필요하지 않을 때 `Apache5HttpClient` 인스턴스를 종료하지 않을 경우 잠재적인 메모리 누수를 방지할 수 있습니다.

다음 예제에서는 S3Client를 생성하고 maxConnections 및 connectionTimeout 값을 `Apache5HttpClient` 사용하여의 임베디드 인스턴스를 구성합니다. HTTP 인스턴스는 `S3Client.Builder`의 `httpClientBuilder` 메서드를 사용하여 생성됩니다.

가져오기

```
import software.amazon.awssdk.http.apache5.Apache5HttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import java.time.Duration;
```

코드

```
S3Client s3Client = S3Client   // Singleton: Use the s3Client for all requests.
    .builder()
    .httpClientBuilder(Apache5HttpClient.builder()
        .maxConnections(100)
        .connectionTimeout(Duration.ofSeconds(5))
    )
    .build();

// Perform work with the s3Client.

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

대안 접근 방식: **Apache5HttpClient** 인스턴스 공유

애플리케이션의 리소스 및 메모리 사용량을 낮추려면 Apache5HttpClient를 구성하고 여러 서비스 클라이언트에서 공유할 수 있습니다. HTTP 연결 풀이 공유되므로 리소스 사용량이 줄어듭니다.

> **Note**
>
> Apache5HttpClient 인스턴스를 공유한 경우 폐기할 준비가 되면 인스턴스를 닫아야 합니다. SDK는 서비스 클라이언트가 닫힐 때 인스턴스를 닫지 않습니다.

다음 예제는 두 서비스 클라이언트에서 사용하는 Apache 기반 HTTP 클라이언트를 구성하는 예제입니다. 구성된 ApacheHttpClient 인스턴스는 각 빌더의 httpClient 메서드로 전달됩니다. 서비스 클라이언트와 HTTP 클라이언트가 더 이상 필요하지 않으면 코드가 명시적으로 닫습니다. 코드는 HTTP 클라이언트를 마지막으로 닫습니다.

가져오기

```
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.apache5.Apache5HttpClient;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.s3.S3Client;
```

코드

```
SdkHttpClient apache5HttpClient = Apache5HttpClient.builder()
        .maxConnections(100).build();

// Singletons: Use the s3Client and dynamoDbClient for all requests.
S3Client s3Client =
    S3Client.builder()
            .httpClient(apache5HttpClient).build();

DynamoDbClient dynamoDbClient =
    DynamoDbClient.builder()
                  .httpClient(apache5HttpClient).build();

// Perform work with the s3Client and dynamoDbClient.

// Requests completed: Close all service clients.
s3Client.close();
dynamoDbClient.close();
apache5HttpClient.close();  // Explicitly close apache5HttpClient.
```

AWS SDK for Java 2.x에서 실행 인터셉터 사용

요청 및 응답 수명 주기에 연결된 AWS SDK for Java 2.x 후크의 실행 인터셉터는 API 호출 실행의 다양한 단계에서 사용자 지정 로직을 수행합니다. 인터셉터를 사용하여 로깅, 지표 수집, 요청 수정, 디버깅 및 오류 처리 등의 크로스 커팅 문제를 구현합니다.

인터셉터는 다양한 요청 실행 단계에 대한 후크를 제공하는 `ExecutionInterceptor` 인터페이스를 구현합니다.

인터셉터 수명 주기

`ExecutionInterceptor` 인터페이스는 요청 실행 중에 특정 시점에서 호출되는 메서드를 제공합니다.

- `beforeExecution` - 요청이 실행되기 전에 호출됩니다.
- `modifyRequest` - SDK 요청 객체를 수정합니다.
- `beforeMarshalling` - HTTP에 대한 요청이 마샬되기 전에 호출됩니다.
- `afterMarshalling` - HTTP에 대한 요청이 마샬된 후에 호출됩니다.
- `modifyHttpRequest` - HTTP 요청을 수정합니다.

- `beforeTransmission` - HTTP 요청이 전송되기 전에 호출됩니다.
- `afterTransmission` - HTTP 응답을 수신한 후에 호출됩니다.
- `modifyHttpResponse` - HTTP 응답을 수정합니다.
- `beforeUnmarshalling` - HTTP 응답이 마샬 해제되기 전에 호출됩니다.
- `afterUnmarshalling` - HTTP 응답이 마샬 해제된 후에 호출됩니다.
- `modifyResponse` - SDK 응답 객체를 수정합니다.
- `afterExecution` - 요청 실행 성공 후에 호출됩니다.
- `onExecutionFailure` - 요청 실행이 실패할 때 호출됩니다.

인터셉터 등록

`overrideConfiguration` 메서드를 사용하여 서비스 클라이언트를 구축할 때 인터셉터를 등록합니다. 여러 인터셉터를 등록할 수 있으며 등록한 순서대로 실행됩니다.

```
// Register a single interceptor.
SqsClient client = SqsClient.builder()
    .overrideConfiguration(c -> c.addExecutionInterceptor(new LoggingInterceptor()))
    .build();

// Register multiple interceptors.
S3Client s3Client = S3Client.builder()
    .overrideConfiguration(config -> config
        .addExecutionInterceptor(new TimingInterceptor())
        .addExecutionInterceptor(new LoggingInterceptor())
        .addExecutionInterceptor(new RequestModificationInterceptor()))
    .build();
```

인터셉터 예제

다음 클래스는 실행 인터셉터를 사용하여 주요 비즈니스 로직을 변경하지 않고 S3 작업에 로깅, 성능 모니터링 및 요청 수정 등의 크로스 커팅 문제를 추가하는 방법을 보여줍니다.

이 예제에서는 S3 클라이언트에 여러 인터셉터를 등록하고 실제 AWS API 호출 중에 해당 인터셉터가 작동하는지 확인하는 방법을 보여줍니다.

가져오기

```
import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.core.ApiName;
import software.amazon.awssdk.core.SdkRequest;
import software.amazon.awssdk.core.interceptor.Context;
import software.amazon.awssdk.core.interceptor.ExecutionAttributes;
import software.amazon.awssdk.core.interceptor.ExecutionInterceptor;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.SdkHttpResponse;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;

import java.time.Duration;
import java.time.Instant;
```

```
public class S3InterceptorsDemo {
    private static final Logger logger =
 LoggerFactory.getLogger(S3InterceptorsDemo.class);

    public static void main(String[] args) {
        logger.info("=== AWS SDK for Java v2 - S3 Interceptors Demo ===");

        // Create an S3 client with multiple interceptors.
        S3Client s3Client = S3Client.builder()
            .overrideConfiguration(config -> config
                .addExecutionInterceptor(new TimingInterceptor())
                .addExecutionInterceptor(new LoggingInterceptor())
                .addExecutionInterceptor(new RequestModificationInterceptor()))
            .build();

        try {
            logger.info("Starting S3 operations with interceptors...");

            // Operation 1: List buckets.
            logger.info("Operation 1: Listing S3 buckets");
            logger.info("---------------------------------------");
            ListBucketsResponse listBucketsResponse = s3Client.listBuckets();
            logger.info("Found {} buckets", listBucketsResponse.buckets().size());

            // Operation 2: Try to access a bucket that likely doesn't exist.
            logger.info("Operation 2: Checking non-existent bucket (demonstrating error
 interceptor)");
```

```
            logger.info("----------------------------------------");
            try {
                s3Client.headBucket(HeadBucketRequest.builder()
                    .bucket("amzn-s3-demo-bucket-that-does-not-exist-1234")
                    .build());
            } catch (Exception e) {
                logger.info("Expected error occurred (interceptor should have logged
it)");
            }

        } catch (Exception e) {
            logger.error(" Error during S3 operations: {}", e.getMessage(), e);
        } finally {
            s3Client.close();
            logger.info("Demo completed - S3 client closed");
        }
    }

    // Logging interceptor.
    private static class LoggingInterceptor implements ExecutionInterceptor {
        private static final Logger logger =
LoggerFactory.getLogger(LoggingInterceptor.class);

        @Override
        public void beforeExecution(Context.BeforeExecution context,
ExecutionAttributes executionAttributes) {
            logger.info("[LOGGING] Starting request: {}",
context.request().getClass().getSimpleName());
        }

        @Override
        public void afterExecution(Context.AfterExecution context, ExecutionAttributes
executionAttributes) {
            logger.info("[LOGGING] Completed request: {}",
context.request().getClass().getSimpleName());
        }

        @Override
        public void onExecutionFailure(Context.FailedExecution context,
ExecutionAttributes executionAttributes) {
            logger.error("[LOGGING] Request failed: {}",
context.request().getClass().getSimpleName());
            if (context.exception() instanceof AwsServiceException) {
                AwsServiceException ase = (AwsServiceException) context.exception();
```

```
                if (ase.awsErrorDetails().errorCode() != null) {
                    SdkHttpResponse httpResponse =
ase.awsErrorDetails().sdkHttpResponse();
                    logger.error("   HTTP Status: {}", httpResponse.statusCode());
                    logger.error("   Error Code: {}",
ase.awsErrorDetails().errorCode());
                    logger.error("   Error Message: {}",
ase.awsErrorDetails().errorMessage());
                }
            }
        }
    }

    // Performance timing interceptor.
    private static class TimingInterceptor implements ExecutionInterceptor {
        private static final Logger logger =
LoggerFactory.getLogger(TimingInterceptor.class);
        private Instant startTime;

        @Override
        public void beforeExecution(Context.BeforeExecution context,
ExecutionAttributes executionAttributes) {
            startTime = Instant.now();
            logger.info("##  [TIMING] Request started at: {}", startTime);
        }

        @Override
        public void afterExecution(Context.AfterExecution context, ExecutionAttributes
executionAttributes) {
            if (startTime != null) {
                Duration duration = Duration.between(startTime, Instant.now());
                logger.info("##  [TIMING] Request completed in: {}ms",
duration.toMillis());
            }
        }

        @Override
        public void onExecutionFailure(Context.FailedExecution context,
ExecutionAttributes executionAttributes) {
            if (startTime != null) {
                Duration duration = Duration.between(startTime, Instant.now());
                logger.warn("##  [TIMING] Request failed after: {}ms",
duration.toMillis());
            }
```

```
        }
    }

    // Request modification interceptor
    private static class RequestModificationInterceptor implements ExecutionInterceptor
{
        private static final Logger logger =
LoggerFactory.getLogger(RequestModificationInterceptor.class);

        @Override
        public SdkRequest modifyRequest(Context.ModifyRequest context,
ExecutionAttributes executionAttributes) {
            SdkRequest originalRequest = context.request();
            logger.info("[MODIFY] Modifying request: {}",
originalRequest.getClass().getSimpleName());

            // For ListBucketsRequest, we can't modify much since it has no settable
properties
            // For HeadBucketRequest, we can demonstrate modifying the request
            if (originalRequest instanceof HeadBucketRequest) {
                HeadBucketRequest headRequest = (HeadBucketRequest) originalRequest;

                // Create a new request with an API name added.
                return HeadBucketRequest.builder()
                        .bucket(headRequest.bucket())
                        .overrideConfiguration(b -> b.addApiName(ApiName.builder()
                                .name("My-API")
                                .version("1.0")
                                .build()))
                        .build();
            }
            logger.info("Not a HeadBucketRequest, returning original request");
            return originalRequest;
        }

        @Override
        public SdkHttpRequest modifyHttpRequest(Context.ModifyHttpRequest context,
ExecutionAttributes executionAttributes) {
            logger.info("[MODIFY] Adding custom HTTP headers");
            return context.httpRequest().toBuilder()
                .putHeader("X-Custom-Header", "S3InterceptorDemo")
                .putHeader("X-Request-ID", java.util.UUID.randomUUID().toString())
                .build();
        }
```

```
    }
}
```

Maven pom 파일

```
<?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>interceptors-examples</artifactId>
    <version>1.0.0</version>
    <packaging>jar</packaging>

    <name>interceptors-examples</name>
    <description>Demonstration of execution interceptors in AWS SDK for Java v2</
description>

    <properties>
        <maven.compiler.source>17</maven.compiler.source>
        <maven.compiler.target>17</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <aws.java.sdk.version>2.31.62</aws.java.sdk.version>
        <exec.mainClass>org.example.S3InterceptorsDemo</exec.mainClass>
    </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>
            <dependency>
                <groupId>org.apache.logging.log4j</groupId>
                <artifactId>log4j-bom</artifactId>
                <version>2.23.1</version>
                <type>pom</type>
```

```
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>s3</artifactId>
        </dependency>

        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-core</artifactId>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
            <version>2.0.13</version>
        </dependency>
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-slf4j2-impl</artifactId>
        </dependency>
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-1.2-api</artifactId>
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter</artifactId>
            <version>5.10.1</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <!-- Compiler Plugin -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.11.0</version>
                <configuration>
```

```
                <source>17</source>
                <target>17</target>
            </configuration>
        </plugin>

        <!-- Surefire Plugin for running tests -->
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-surefire-plugin</artifactId>
            <version>3.2.2</version>
        </plugin>

        <!-- Exec Plugin for running the main class -->
        <plugin>
            <groupId>org.codehaus.mojo</groupId>
            <artifactId>exec-maven-plugin</artifactId>
            <version>3.1.0</version>
            <configuration>
                <mainClass>${exec.mainClass}</mainClass>
            </configuration>
        </plugin>

        <!-- Shade Plugin to create executable JAR -->
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-shade-plugin</artifactId>
            <version>3.4.1</version>
            <executions>
                <execution>
                    <phase>package</phase>
                    <goals>
                        <goal>shade</goal>
                    </goals>
                    <configuration>
                        <transformers>
                            <transformer
 implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">
                                <mainClass>${exec.mainClass}</mainClass>
                            </transformer>
                        </transformers>
                        <createDependencyReducedPom>false</
createDependencyReducedPom>
                    </configuration>
                </execution>
```

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

모범 사례

- 인터셉터를 가볍게 유지 - 인터셉터는 각 요청에 대해 실행되므로 성능에 영향을 미칠 수 있는 과도한 계산이나 차단 작업은 지양합니다.
- 예외 정상 처리 - 인터셉터에서 예외가 발생하면 전체 요청이 실패합니다. 작업 실패가 발생할 확률이 있는 경우 항상 try-catch 블록을 사용합니다.
- 순서 중요 - 인터셉터는 등록한 순서대로 실행됩니다. 인터셉터를 등록할 때 인터셉터 간의 종속성을 고려합니다.
- 상태에 ExecutionAttributes 사용 - 서로 다른 인터셉터 메서드 간에 데이터를 전달해야 하는 경우 인스턴스 변수 대신 `ExecutionAttributes`를 사용하여 스레드 안전을 확보합니다.
- 민감한 데이터에 유의 - 요청 및 응답을 로그할 때 자격 증명 또는 개인정보와 같은 민감한 정보를 로그하지 않도록 주의합니다.

컨텍스트 객체

각 인터셉터 메서드는 다양한 실행 단계에서 요청 및 응답 정보에 대한 액세스를 제공하는 컨텍스트 객체를 수신합니다.

- `Context.BeforeExecution` - 원래 SDK 요청에 대한 액세스 권한을 제공합니다.
- `Context.ModifyRequest` - SDK 요청을 수정합니다.
- `Context.ModifyHttpRequest` - HTTP 요청을 수정합니다.
- `Context.AfterExecution` - 요청 및 응답 모두에 대한 액세스 권한을 제공합니다.
- `Context.FailedExecution` - 요청 및 발생한 예외에 대한 액세스 권한을 제공합니다.

AWS SDK for Java 2.x 사용

이 장에서는 AWS SDK for Java 2.x를 효과적으로 사용하는 방법을 보여줍니다. 서비스 클라이언트를 만들고, 요청을 만들고, 응답을 처리하고, 오류를 관리하는 방법을 알아봅니다. 이 장에서는 동기식 및 비동기 프로그래밍, 페이지 매김 결과, 리소스 모니터링을 위한 웨이터 및 성능 최적화를 다룹니다.

클라이언트 재사용, 문제 해결 가이드, Lambda 시작 최적화, HTTP/2 지원 및 DNS 구성에 대한 모범 사례도 확인할 수 있습니다.

목차

- JVM TTL을 설정하는 방법
- AWS SDK for Java에서 HTTP/2 작업

AWS SDK for Java 2.x를 사용하여 AWS 서비스 요청하기

서비스 클라이언트를 사용하여 요청 만들기

SDK 설정의 단계를 완료하고 서비스 클라이언트를 구성하는 방법을 알면 Amazon S3, Amazon DynamoDB, AWS Identity and Access Management, Amazon EC2 등의AWS 서비스에 요청할 준비가 된 것입니다.

서비스 클라이언트 생성

AWS 서비스에 요청하려면 먼저 정적 팩토리 메서드인 `builder()`를 사용하여 해당 서비스의 서비스 클라이언트를 인스턴스화해야 합니다. 이 `builder()` 메서드는 서비스 클라이언트를 사용자 지정할 수 있는 `builder` 객체를 반환합니다. 유용한 setter 메서드는 `builder` 객체를 반환해 읽기 쉽고 편리하도록 메서드 호출을 묶을 수 있게 도와줍니다. 원하는 속성을 구성한 후에는 `build()` 메서드를 호출하여 클라이언트를 생성할 수 있습니다.

예를 들어, 이 코드 조각은 `Ec2Client` 객체를 Amazon EC2용 서비스 클라이언트로 인스턴스화합니다.

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

> **Note**
>
> SDK의 서비스 클라이언트는 스레드 세이프입니다. 최상의 성능을 위해 수명이 긴 객체로 처리합니다. 각 클라이언트마다 클라이언트에서 가비지가 수집될 때 릴리스되는 고유한 연결 풀 리소스가 있습니다.
> 서비스 클라이언트 객체는 변경할 수 없으므로 요청하는 각 서비스에 대해 또는 동일한 서비스에 대한 요청을 위해 다른 구성을 사용하려는 경우 새 클라이언트를 만들어야 합니다.
> 모든 AWS 서비스에 대해 서비스 클라이언트 빌더에서 `Region`를 지정할 필요는 없지만 애플리케이션에서 수행하는 API 호출에 대해 리전을 설정하는 것이 가장 좋습니다. 자세한 내용은 AWS 리전 선택을 참조하세요.

기본 클라이언트 구성

클라이언트 빌더에는 `create()`라는 또 다른 팩토리 메서드가 있습니다. 기본 구성을 사용해 서비스 클라이언트를 생성하는 메서드입니다. 기본 공급자 체인을 사용하여 자격 증명과 기본값 AWS 리전 공급자 체인을 로드합니다. 애플리케이션이 실행되는 환경에서 자격 증명이나 리전을 확인할 수 없는 경우 `create` 에 대한 호출이 실패합니다. SDK가 사용할 자격 증명 및 리전을 결정하는 방법에 대한 자세한 내용은 자격 증명 사용 및 리전 선택을 참조하세요.

예를 들어, 이 코드 조각은 `DynamoDbClient` 객체를 Amazon DynamoDB용 서비스 클라이언트로 인스턴스화합니다.

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

서비스 클라이언트 구성

서비스 클라이언트를 구성하는 방법에 대한 자세한 내용은 외부에서 클라이언트 구성 및 코드의 클라이언트 구성 섹션을 참조하세요.

서비스 클라이언트 닫기

가장 좋은 방법은 애플리케이션 수명 동안 여러 API 서비스 호출에 서비스 클라이언트를 사용하는 것입니다. 하지만 일회용으로 사용할 서비스 클라이언트가 필요하거나 더 이상 필요 없는 서비스 클라이언트는 닫으세요.

리소스를 확보하는 데 서비스 클라이언트가 더 이상 필요하지 않을 때 `close()` 메서드를 호출하세요.

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

일회용으로 사용할 서비스 클라이언트가 필요한 경우 `try`-with-resources 문을 사용하여 서비스 클라이언트를 리소스로 인스턴스화할 수 있습니다. 서비스 클라이언트가 `Autoclosable` 인터페이스를 구현하므로 JDK는 명령문 끝에서 `close()` 메서드를 자동으로 호출합니다.

다음 예제에서는 서비스 클라이언트를 사용하여 일회성 호출을 실행하는 방법을 보여줍니다. AWS Security Token Service를 호출하는 `StsClient`는 계정 ID를 반환한 후 종료됩니다.

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

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

```
    }
}
```

요청을 생성

서비스 클라이언트를 사용하여 해당 AWS 서비스에 요청합니다.

예를 들어, 이 코드 조각은 `RunInstancesRequest` 객체를 생성하여 새 Amazon EC2 인스턴스를 생성하는 방법을 보여줍니다.

```
// 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);
```

요청을 만들고 인스턴스를 전달하는 대신 SDK는 요청을 만드는 데 사용할 수 있는 유용한 API를 제공합니다. 유용한 API를 사용하면 Java lambda 표현식을 사용하여 요청을 '인라인'으로 만들 수 있습니다.

다음 예제에서는 빌더를 사용하여 요청을 생성하는 `runInstances` 메서드의 버전을 사용하여 이전 예제를 다시 작성합니다.

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

요청을 사용하여 클라이언트 구성 재정의

서비스 클라이언트는 변경할 수 없지만 요청 수준에서 많은 해당 설정을 재정의할 수 있습니다. 요청을 구축할 때 AwsRequestOverrideConfiguration 인스턴스를 제공하여 재정의된 설정을 제공할 수 있습니다. 클라이언트 설정을 재정의하는 데 사용할 수 있는 몇 가지 방법은 다음과 같습니다.

- apiCallAttemptTimeout
- apiCallTimeout
- credentialProvider
- compressionConfiguration
- putHeader

클라이언트 설정을 요청으로 재정의하는 예제의 경우 기본 설정을 사용하는 다음 S3 클라이언트가 있다고 가정합니다.

```
S3Client s3Client = S3Client.create();
```

대용량 파일을 다운로드하고 다운로드가 완료되기 전에 요청의 제한 시간이 초과되지 않도록 하려고 합니다. 이렇게 하려면 다음 코드와 같이 단일 GetObject 요청의 제한 시간 값만 늘립니다.

Standard API

```
AwsRequestOverrideConfiguration overrideConfiguration =
 AwsRequestOverrideConfiguration.builder()
    .apiCallTimeout(Duration.ofSeconds(100L))
    .apiCallAttemptTimeout(Duration.ofSeconds(25L))
    .build();

GetObjectRequest request = GetObjectRequest.builder()
    .bucket("amzn-s3-demo-bucket")
    .key("demo-key")
    .overrideConfiguration(overrideConfiguration)
    .build();

s3Client.getObject(request, myPath);
```

Fluent API

```
s3Client.getObject(b -> b
        .bucket("amzn-s3-demo-bucket")
        .key("demo-key")
        .overrideConfiguration(c -> c
            .apiCallTimeout(Duration.ofSeconds(100L))
            .apiCallAttemptTimeout(Duration.ofSeconds(25L))),
    myPath);
```

응답 처리

SDK는 대부분의 서비스 작업에 대한 응답 객체를 반환합니다. 코드는 필요에 따라 응답 객체의 정보를 처리할 수 있습니다.

예를 들어 다음 코드 조각은 이전 요청에서 `RunInstancesResponse` 객체와 함께 반환된 첫 번째 인스턴스 ID를 출력합니다.

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

그러나 일부 작업은 서비스별 데이터가 포함된 응답 객체를 반환하지 않습니다. 이러한 상황에서는 HTTP 응답 상태를 쿼리하여 작업이 성공했는지 확인할 수 있습니다.

예를 들어 다음 코드 조각의 코드는 HTTP 응답을 확인하여 Amazon Simple Email Service의 `DeleteContactList` 작업이 성공했는지 확인합니다.

```
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());
}
```

AWS SDK for Java 2.x를 사용한 비동기식 프로그래밍 작업

AWS SDK for Java 2.x는 일부 스레드에 높은 동시성을 구현하는 비차단형 입출력 지원이 포함된 비동기식 클라이언트가 특징입니다. 그러나 전체 비차단 입출력은 보장되지 않습니다. 비동기식 클라이언트는 자격 증명 검색, AWS Signature Version 4(SigV4)를 사용한 요청 서명 또는 엔드포인트 검색과 같은 경우에 차단 호출을 수행할 수 있습니다.

비동기 메서드는 클라이언트가 서비스로부터 응답을 받을 때까지 스레드의 실행을 차단합니다. 비동기 메서드는 (값을) 즉시 반환하며, 응답을 기다리지 않고 제어 권한을 호출 스레드에 넘겨줍니다.

비동기 메서드는 응답이 제공되기 전에 (값을) 반환하므로 준비되었을 때 응답을 가져올 방법이 필요합니다. AWS SDK for Java의 2.x에서 비동기식 클라이언트 메서드는 준비가 되었을 때 응답에 액세스하도록 허용하는 CompletableFuture 객체를 반환합니다.

비동기식 클라이언트 API 사용

비동기식 클라이언트 메서드의 서명은 동기식 클라이언트 메서드와 동일하지만 비동기식 메서드는 향후 비동기식 작업의 결과가 포함된 CompletableFuture 객체를 반환합니다. SDK의 비동기식 메서드가 실행되는 동안 오류가 발생하면 오류가 `CompletionException`으로 발생합니다.

결과를 얻는 데 사용할 수 있는 한 가지 접근 방식은 SDK 메서드 호출에서 반환된 `whenComplete()`에 `CompletableFuture` 메서드를 연결하는 것입니다. `whenComplete()` 메서드는 비동기식 호출이 완료된 방식에 따라 결과 또는 `CompletionException` 유형의 Throwable 객체를 수신합니다. 호출 코드로 반환되기 전에 결과를 처리하거나 확인하는 작업을 `whenComplete()`에 제공합니다.

SDK 메서드에서 반환한 객체 이외의 다른 것을 반환하려면 `handle()` 메서드를 대신 사용합니다. `handle()` 메서드는 `whenComplete()`와 동일한 파라미터를 사용하지만 결과를 처리하고 객체를 반환할 수 있습니다.

비동기식 체인이 완료되고 완료 결과를 검색할 때까지 기다리기 위해 `join()` 메서드를 호출할 수 있습니다. `Throwable` 객체가 체인에서 처리되지 않은 경우 `join()` 메서드는 원래 예외를 래핑하는 선택되지 않은`CompletionException`을 발생시킵니다. `CompletionException#getCause()`를 사용하여 원래 예외에 액세스합니다. 또한 `CompletableFuture#get()` 메서드를 호출하여 완료 결과를 가져올 수 있습니다. 그러나 `get()` 메서드는 확인된 예외를 발생시킬 수 있습니다.

다음 예제에서는 DynamoDB 비동기식 클라이언트의 `listTables()` 메서드를 사용하는 방법에 대한 2가지 변형을 보여줍니다. `whenComplete()`에 전달된 작업은 성공적인 응답을 로깅하는 반면, `handle()` 버전은 테이블 이름 목록을 추출하고 목록을 반환합니다. 두 경우 모두 비동기식 체인에서 오류가 생성되는 경우 클라이언트 코드가 오류를 처리할 수 있도록 오류를 다시 발생시킵니다.

가져옵니다.

```
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;
```

코드

whenComplete() variation

```
public class DynamoDbAsyncListTables {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        DynamoDbAsyncClient dynamoDbAsyncClient =
 DynamoDbAsyncClient.builder().region(region).build();
        try {
            ListTablesResponse listTablesResponse =
 listTablesWhenComplete(dynamoDbAsyncClient).join();  // The join() method may throw
 a CompletionException.
            if (listTablesResponse.hasTableNames()){
                System.out.println("Table exist in this region: " + region.id());
            }
        } catch (RuntimeException e) {
            // Handle as needed. Here we simply print out the class names.
            System.out.println(e.getClass()); // Prints 'class
 java.util.concurrent.CompletionException'.
            System.out.println(e.getCause().getClass()); // Prints 'class
 software.amazon.awssdk.services.dynamodb.model.DynamoDbException'.
        }
    }

    public static CompletableFuture<ListTablesResponse>
 listTablesWhenComplete(DynamoDbAsyncClient client) {
        return client.listTables(ListTablesRequest.builder().build())
            .whenComplete((listTablesResponse, throwable) -> {
                if (listTablesResponse != null) {  // Consume the response.
                    System.out.println("The SDK's listTables method completed
 successfully.");
                } else {
                    RuntimeException cause = (RuntimeException)
 throwable.getCause(); // If an error was thrown during the SDK's listTables method
 it is wrapped in a CompletionException.

  // The SDK throws only RuntimeExceptions, so this is a safe cast.
                    System.out.println(cause.getMessage());  // Log error here, but
 rethrow so the calling code can handle as needed.
                    throw cause;
                }
            });
```

```
    }
```

handle() variation

```
public class DynamoDbAsyncListTables {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        DynamoDbAsyncClient dynamoDbAsyncClient =
 DynamoDbAsyncClient.builder().region(region).build();
        try {
            List<String> tableNames =
 listTablesHandle(dynamoDbAsyncClient).join(); // The join() method may throw a
 CompletionException.
            tableNames.forEach(System.out::println);
        } catch (RuntimeException e) {
            // Handle as needed. Here we simply print out the class names.
            System.out.println(e.getClass()); // Prints 'class
 java.util.concurrent.CompletionException'.
            System.out.println(e.getCause().getClass()); // Prints 'class
 software.amazon.awssdk.services.dynamodb.model.DynamoDbException'.
        }
    }

    public static CompletableFuture<List<String>>
 listTablesHandle(DynamoDbAsyncClient client) {
        return client.listTables(ListTablesRequest.builder().build())
            .handle((listTablesResponse, throwable) -> {
                if (listTablesResponse != null) {
                    return listTablesResponse.tableNames(); // Return the list of
 table names.
                } else {
                    RuntimeException cause = (RuntimeException)
 throwable.getCause(); // If an error was thrown during the SDK's listTables method
 it is wrapped in a CompletionException.

  // The SDK throws only RuntimeExceptions, so this is a safe cast.
                    System.out.println(cause.getMessage());  // Log error here, but
 rethrow so the calling code can handle as needed.
                    throw cause;
                }
            });
    }
```

```
}
```

비동기식 메서드에서 스트리밍 처리

스트리밍 콘텐츠에서 비동기식 메서드를 사용할 경우 증분 방식으로 콘텐츠를 제공하기 위해 AsyncRequestBody를 제공하거나, 응답을 수신해 처리할 수 있도록 AsyncResponseTransformer를 제공해야 합니다.

다음 예제는 `PutObject` 작업의 비동기식 양식을 사용하여 파일을 Amazon S3에 비동기식으로 업로드합니다.

가져옵니다.

```
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;
```

코드

```
/**
 * 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();
```

```
    }
}
```

다음 예제는 Amazon S3에서 `GetObject` 작업의 비동기식 양식을 사용해 파일을 가져옵니다.

가져옵니다.

```
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;
```

코드

```
/**
 * 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();
    }
}
```

고급 비동기 옵션 구성

AWS SDK for Java 2.x는 비동기식 이벤트 중심 네트워크 애플리케이션 프레임워크인 Netty를 사용하여 I/O 스레드를 처리합니다. AWS SDK for Java 2.x는 Netty 뒤에 `ExecutorService`를 생성하여 HTTP 클라이언트 요청에서 Netty 클라이언트로 반환된 선물을 완료합니다. 이러한 추상화는 개발자가 스레드 중지 또는 대기를 선택하는 경우 애플리케이션이 비동기 프로세스를 차단하는

위험을 줄여줍니다. 기본적으로 각 비동기 클라이언트는 프로세서 수에 따라 스레드 풀을 만들고 ExecutorService 내에서 대기열의 작업을 관리합니다.

비동기식 클라이언트를 구축할 경우 ExecutorService의 특정 JDK 구현을 지정할 수 있습니다. 다음 코드 조각은 고정된 수의 스레드로 ExecutorService를 만듭니다.

코드

```
S3AsyncClient clientThread = S3AsyncClient.builder()
  .asyncConfiguration(
    b -> b.advancedOption(SdkAdvancedAsyncClientOption
      .FUTURE_COMPLETION_EXECUTOR,
      Executors.newFixedThreadPool(10)
    )
  )
  .build();
```

성능을 최적화하려면 자체 스레드 풀 실행 도구를 관리하고 클라이언트를 구성할 때 이를 포함시킬 수 있습니다.

```
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();
```

AWS SDK for Java 2.x 사용 모범 사례

API 제한 시간을 구성하여 중단 요청 방지

SDK는 연결 제한 시간 및 소켓 제한 시간과 같은 일부 제한 시간 옵션에 대한 기본값을 제공하지만 API 호출 제한 시간이나 개별 API 호출 시도 제한 시간에 대해서는 기본값을 제공하지 않습니다. 개별 시도와 전체 요청 모두에 대해 제한 시간을 설정하는 것이 좋습니다. 이렇게 하면 요청 시도를 완료하는 데 시간이 더 오래 걸리거나 치명적인 네트워크 문제를 일으킬 수 있는 일시적인 문제가 있는 경우 최적의 방식으로 애플리케이션이 빠르게 실패하도록 보장합니다.

ClientOverrideConfiguration#apiCallAttemptTimeout 및 ClientOverrideConfiguration#apiCallTimeout를 사용하여 서비스 클라이언트의 모든 요청에 대한 제한 시간을 구성할 수 있습니다.

다음 예제는 사용자 지정 시간 제한 값을 사용하는 Amazon S3 클라이언트의 구성을 보여줍니다.

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

apiCallAttemptTimeout

이 설정은 단일 HTTP 시도 시간을 설정합니다. 이 시간이 지나면 API 호출을 재시도할 수 있습니다.

apiCallTimeout

이 속성의 값은 모든 재시도를 포함하여 전체 실행에 걸리는 시간을 구성합니다.

서비스 클라이언트에서 이러한 제한 시간 값을 설정하는 대신 RequestOverrideConfiguration#apiCallTimeout() 및 RequestOverrideConfiguration#apiCallAttemptTimeout()를 사용하여 단일 요청을 구성할 수 있습니다.

다음 예시에서는 사용자 지정 제한 시간 값을 사용하여 단일 `listBuckets` 요청을 구성합니다.

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

```
                .apiCallAttemptTimeout(Duration.ofMillis(<custom value>))));
```

이러한 속성을 함께 사용하면 재시도 전체에 걸쳐 모든 시도에 소요되는 총 시간에 대한 엄격한 제한을 설정합니다. 또한 느린 요청에 대해 빠르게 실패하도록 개별 HTTP 요청을 설정합니다.

서비스 클라이언트를 재사용하여 성능 향상

각 서비스 클라이언트는 자체 HTTP 연결 풀을 유지합니다. 풀에 이미 있는 연결을 새 요청에 재사용하여 새 연결을 설정하는 시간을 줄일 수 있습니다. 효과적으로 사용되지 않는 연결 풀이 너무 많아서 발생하는 오버헤드를 방지하려면 클라이언트의 단일 인스턴스를 공유하는 것이 좋습니다. 모든 서비스 클라이언트는 스레드로부터 안전합니다.

클라이언트 인스턴스를 공유하지 않으려면 클라이언트가 필요하지 않을 때 인스턴스를 `close()` 호출하여 리소스를 릴리스하세요.

미사용 서비스 클라이언트를 닫아 리소스 누수 방지

더 이상 필요하지 않은 경우 서비스 클라이언트를 종료해 스레드 등의 리소스를 릴리스합니다. 클라이언트 인스턴스를 공유하지 않으려면 클라이언트가 필요하지 않을 때 인스턴스를 `close()` 호출하여 리소스를 릴리스하세요.

입력 스트림을 닫아 연결 풀 소진 방지

`ResponseInputStream`를 사용하여 직접 작업하는 경우 `S3Client#getObject`와 같은 스트리밍 작업의 경우 다음을 수행하는 것이 좋습니다.

- 가능한 한 빨리 입력 스트림에서 모든 데이터를 읽습니다.
- 가능한 한 빨리 입력 스트림을 닫습니다.

입력 스트림은 HTTP 연결의 데이터에 대한 직접적인 스트림이고 스트림의 모든 데이터를 읽고 스트림을 닫을 때까지 기본 HTTP 연결을 재사용할 수 없기 때문에 이러한 권장 사항을 제시합니다. 이러한 규칙을 따르지 않으면 클라이언트는 열려 있지만 사용되지 않는 HTTP 연결을 너무 많이 할당하여 리소스가 부족해질 수 있습니다.

애플리케이션 워크로드에 대한 HTTP 성능 최적화

SDK는 일반 사용 사례에 적용되는 기본 http 구성 세트를 제공합니다. 고객은 사용 사례에 따라 애플리케이션에 대한 HTTP 구성을 조정하는 것이 좋습니다.

좋은 출발점으로 SDK는 스마트 구성 기본값 기능을 제공합니다. 이 기능은 버전 2.17.102부터 사용할 수 있습니다. 사용 사례에 따라 적절한 구성 값을 제공하는 모드를 선택합니다.

비동기식 클라이언트용 OpenSSL로 SSL 성능 개선

기본적으로 SDK의 `NettyNioAsyncHttpClient`는 JDK의 기본 SSL 구현을 `SslProvider`과 같이 사용합니다. 테스트 결과 OpenSSL이 JDK의 기본 구현보다 성능이 더 좋은 것으로 나타났습니다. Netty 커뮤니티에서도 OpenSSL 사용을 권장합니다.

OpenSSL을 사용하려면 종속성에 `netty-tcnative`을 추가하세요. 구성 세부 정보는 Netty 프로젝트 설명서를 참조하세요.

프로젝트에 맞게 `netty-tcnative`를 구성한 후 `NettyNioAsyncHttpClient` 인스턴스는 자동으로 OpenSSL을 선택합니다. 또는 다음 코드 조각과 같이 `NettyNioAsyncHttpClient` 빌더를 사용하여 `SslProvider`를 명시적으로 설정할 수 있습니다.

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

SDK 지표를 사용하여 애플리케이션 성능 모니터링

Java용 SDK는 애플리케이션의 서비스 클라이언트에 대한 메트릭을 수집할 수 있습니다. 이러한 지표를 사용하여 성능 문제를 식별하고, 전반적인 사용 추세를 검토하고, 반환된 서비스 클라이언트 예외를 검토하거나, 특정 문제를 이해하기 위해 자세히 알아볼 수 있습니다.

애플리케이션 성능을 더 깊이 이해하려면 지표를 수집한 다음 Amazon CloudWatch Logs를 분석하는 것이 좋습니다.

AWS SDK for Java 2.x에서 오류 처리

AWS SDK for Java 2.x에서 예외가 언제 어떻게 발생하는지를 이해하는 것은 SDK를 사용하여 고품질의 애플리케이션을 빌드하는 데 있어서 중요합니다. 다음 단원에서는 SDK에서 발생하는 다양한 예외의 경우와 이러한 예외를 적절히 처리하는 방법에 대해 설명합니다.

확인되지 않은 예외가 발생하는 이유

AWS SDK for Java에서는 다음과 같은 이유로 확인된 예외 대신에 실행시간 (또는 확인되지 않은) 예외를 사용합니다.

- 개발자가 중요하지 않은 예외 경우를 강제로 처리하지 않고 (또한 해당 코드를 상세 표시 모드로 설정하지 않고) 처리하고자 하는 오류에 대해서만 세부적으로 제어할 수 있도록 하기 위해
- 대규모 애플리케이션에서 확인된 예외 고유의 확장성 문제를 방지하기 위해

일반적으로 확인된 예외는 소규모 애플리케이션에서 잘 작동하는 편이지만, 애플리케이션이 확장되고 복잡해짐에 따라 문제가 될 수도 있습니다.

AwsServiceException(및 하위 클래스)

AwsServiceException은 AWS SDK for Java를 사용할 때 발생하는 가장 일반적인 예외입니다. `AwsServiceException`는 보다 일반적인 SdkServiceException의 서브클래스입니다. `AwsServiceException`는 AWS 서비스의 오류 응답을 나타냅니다. 예를 들어 존재하지 않는 Amazon EC2 인스턴스를 종료하려고 할 경우 Amazon EC2는 오류 응답을 반환하며 해당 오류 응답에 대한 모든 세부 정보가 발생된 `AwsServiceException`에 포함됩니다.

`AwsServiceException`이 발생하면 요청이 AWS 서비스로 전송되었지만 처리되지 못했음을 의미합니다. 이는 요청의 파라미터 오류 또는 서비스 측의 문제로 인해 발생할 수 있습니다.

`AwsServiceException`은 다음과 같은 정보를 제공합니다.

- 반환된 HTTP 상태 코드
- 반환된 AWS 오류 코드
- AWSerrorDetails 클래스에 있는 서비스의 자세한 오류 메시지
- 실패한 요청의 AWS 요청 ID

경우에 따라서는 개발자가 catch 블록을 통해 오류 경우 처리를 세부적으로 제어할 수 있도록 하기 위해 `AwsServiceException`의 하위 클래스가 발생하기도 합니다. AwsServiceException에 대한 Java SDK API 참조에는 많은 수의 `AwsServiceException` 서브클래스가 표시됩니다. 서브클래스 링크를 사용하여 서비스에서 발생하는 세분화된 예외를 자세히 확인할 수 있습니다.

예를 들어 SDK API 참조에 대한 다음 링크는 몇 가지 일반적인 AWS 서비스에 대한 예외 계층 구조를 보여줍니다. 각 페이지에 표시된 서브클래스 목록은 코드가 포착할 수 있는 특정 예외를 보여줍니다.

- Amazon S3()
- DynamoDB
- Amazon SQS

예외에 대해 자세히 알아보려면 AwsErrorDetails 객체의 errorCode를 살펴보세요. 이 errorCode 값을 사용하여 서비스 가이드 API에서 정보를 조회할 수 있습니다. 예를 들어 S3Exception가 InvalidRequest 발견되었는데 AwsErrorDetails#errorCode() 값이 인 경우 Amazon S3 API 참조의 오류 코드 목록을 사용하여 자세한 내용을 확인하세요.

SdkClientException

SdkClientException은 AWS로 요청을 전송하려 하거나 AWS의 응답을 파싱하려 하는 동안 Java 클라이언트 코드 내부에서 문제가 발생했음을 나타냅니다. SdkClientException은 일반적으로 SdkServiceException보다 더 심각하며, 클라이언트에서 AWS 서비스를 호출할 수 없게 하는 중요한 문제를 나타냅니다. 예를 들어 클라이언트 중 하나에서 작업을 호출하려 할 때 네트워크 연결을 사용할 수 없는 경우AWS SDK for Java는 SdkClientException을 발생합니다.

예외 및 재시도 동작

Java용 SDK는 몇 가지 클라이언트측 예외 및 AWS 서비스 응답에서 수신한 HTTP 상태 코드에 대한 요청을 재시도합니다. 이러한 오류는 서비스 클라이언트가 기본적으로 RetryMode 사용하는 레거시의 일부로 처리됩니다. 의 Java API 참조에서는 모드를 구성할 수 있는 다양한 방법을 RetryMode 설명합니다.

자동 재시도를 트리거하는 예외 및 HTTP 상태 코드를 사용자 정의하려면 RetryOnExceptionsCondition 및 RetryOnStatusCodeCondition 인스턴스를 추가하는 RetryPolicy와 함께 서비스 클라이언트를 구성하세요.

AWS SDK for Java 2.x의 페이지가 매겨진 결과 사용

많은 AWS 작업이 응답 객체가 너무 커서 단일 응답을 반환할 수 없을 때 페이지 매김 결과를 반환합니다. AWS SDK for Java 1.0에서는 응답에 다음 결과 페이지를 가져오는 데 사용하는 토큰이 포함됩니다. 대조적으로, AWS SDK for Java 2.x에는 자동으로 다음 결과 페이지를 얻기 위해 여러 서비스 호출을 수행하는 자동 페이지 매김 메서드가 있습니다. 결과를 처리할 코드를 작성하기만 하면 됩니다. 자동 페이지 매김은 동기 클라이언트와 비동기 클라이언트 모두에서 사용할 수 있습니다.

> **Note**
>
> 이 코드 조각은 사용자가 SDK 사용의 기본 사항을 이해하고 Single Sign-On 액세스로 환경을 구성했다고 가정합니다.

동기식 페이지 매김

다음 예제에서는 Amazon S3 버킷의 객체를 나열하는 동기식 페이지 매김 방법을 보여줍니다.

페이지 반복

첫 번째 예제는 `listRes` 페이지네이터 객체인 `ListObjectsV2Iterable` 인스턴스를 사용하여 `stream` 메서드로 모든 응답 페이지를 반복하는 방법을 보여줍니다. 코드는 응답 페이지를 통해 스트리밍하고 응답 스트림을 `S3Object` 콘텐츠 스트림으로 변환한 다음 Amazon S3 객체의 콘텐츠를 처리합니다.

다음 가져오기는 이 동기 페이지 매김 단원의 모든 예제에 적용됩니다.

가져오기

```
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()));
```

GitHub의 전체 예제를 참조하세요.

객체 반복

다음은 응답 페이지 대신 응답에서 반환된 객체에 대해 반복하는 방법을 설명한 예입니다. `ListObjectsV2Iterable`클래스의 `contents` 메서드는 기본 콘텐츠 요소를 처리하기 위한 여러 메서드를 제공하는 `SdkIterable`을 반환합니다.

스트림 사용

다음 코드 조각은 응답 내용에 대해 `stream` 메서드를 사용하여 페이지 매김 항목 모음에 대해 반복합니다.

```
        // Helper method to work with paginated collection of items directly.
        listRes.contents().stream()
            .forEach(content -> System.out
                .println(" Key: " + content.key() + " size = " + content.size()));
```

GitHub의 전체 예제를 참조하세요.

for-each 루프를 사용

`SdkIterable`는 `Iterable` 인터페이스를 확장하므로 내용을 `Iterable`과 같이 처리할 수 있습니다. 다음 코드 조각은 표준 `for-each` 루프를 사용하여 응답 내용을 반복합니다.

```
        for (S3Object content : listRes.contents()) {
```

```
            System.out.println(" Key: " + content.key() + " size = " + content.size());
        }
```

GitHub의 전체 예제를 참조하세요.

수동 페이지 매김

사용 사례에 따라 필요한 경우 수동 페이지 매김을 사용할 수도 있습니다. 후속 요청에 응답 객체의 다음 토큰을 사용합니다. 다음 예에는 `while` 루프가 사용됩니다.

```
        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();
        }
```

GitHub의 전체 예제를 참조하세요.

비동기 페이지 매김

다음 예제는 DynamoDB 테이블을 나열하는 비동기 페이지 매김 방법을 보여줍니다.

테이블 이름 페이지 반복

다음 두 예제는 `ListTablesPublisher`를 가져오기 위한 요청과 함께 `listTablesPaginator` 메서드를 호출하는 비동기식 DynamoDB 클라이언트를 사용합니다. `ListTablesPublisher`는 응답을

처리하기 위한 많은 옵션을 제공하는 2개의 인터페이스를 구현합니다. 각 인터페이스의 메서드를 살펴보겠습니다.

`Subscriber` 사용

다음 코드 예제에서는 `ListTablesPublisher`에서 구현된 `org.reactivestreams.Publisher` 인터페이스를 사용하여 페이지가 매겨진 결과를 처리하는 방법을 보여줍니다. 반응적 스트림 모델에 대한 자세한 내용은 반응적 스트림 GitHub 리포지토리를 참조하세요.

다음 가져오기는 이 비동기 페이지 매김 단원의 모든 예제에 적용됩니다.

가져오기

```
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;
```

다음 코드는 `ListTablesPublisher` 인스턴스를 획득합니다.

```
        // 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);
```

다음 코드는 `org.reactivestreams.Subscriber`의 익명 구현을 사용하여 각 페이지의 결과를 처리합니다.

onSubscribe 메서드가 Subscription.request 메서드를 호출해 게시자에게 데이터를 요청하기 시작합니다. 게시자에게 데이터를 가져오기 시작할 때 호출해야 하는 메서드입니다.

구독자의 onNext 메서드는 모든 테이블 이름에 액세스하고 각 이름을 인쇄하여 응답 페이지를 처리합니다. 페이지가 처리된 후 게시자에게 다른 페이지가 요청됩니다. 모든 페이지를 검색할 때까지 반복적으로 호출되는 메서드입니다.

데이터 검색 동안 오류가 발생할 경우 onError 메서드가 트리거됩니다. 마지막으로 모든 페이지를 다 요청했을 때 onComplete 메서드가 호출됩니다.

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

GitHub의 전체 예제를 참조하세요.

`Consumer` 사용

`ListTablesPublisher`가 구현하는 `SdkPublisher` 인터페이스에는 `Consumer`를 받아서 `CompletableFuture<Void>`를 반환하는 `subscribe` 메서드가 있습니다.

이 인터페이스의 `subscribe` 메서드는 `org.reactivestreams.Subscriber`의 오버헤드가 너무 클 때 간단한 사용 사례에 사용할 수 있습니다. 아래 코드는 각 페이지를 사용하므로 각 페이지에서 `tableNames` 메서드를 호출합니다. `tableNames` 메서드는 `forEach` 메서드로 처리된 DynamoDB 테이블 이름 중 `java.util.List`를 반환합니다.

```
        // Use a Consumer for simple use cases.
        CompletableFuture<Void> future = publisher.subscribe(
                response -> response.tableNames()
                        .forEach(System.out::println));
```

GitHub의 전체 예제를 참조하세요.

테이블 이름 반복

다음은 응답 페이지 대신 응답에서 반환된 객체에 대해 반복하는 방법을 설명한 예입니다. 이전에 해당 `contents` 메서드와 함께 표시된 동기 Amazon S3 예제와 마찬가지로, DynamoDB 비동기 결과 클래스 `ListTablesPublisher`에는 기본 항목 컬렉션과 상호 작용할 수 있는 편리한 `tableNames` 메서드가 있습니다. `tableNames` 메서드의 반환 유형은 모든 페이지에서 항목을 요청하기 위해 사용할 수 있는 `SdkPublisher`입니다.

`Subscriber` 사용

다음 코드는 테이블 이름의 기본 컬렉션 중 `SdkPublisher`를 가져옵니다.

```
        // 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();
```

다음 코드는 `org.reactivestreams.Subscriber`의 익명 구현을 사용하여 각 페이지의 결과를 처리합니다.

구독자의 `onNext` 메서드는 컬렉션의 개별 요소를 처리합니다. 이 경우에는 테이블 이름입니다. 테이블 이름이 처리된 후 게시자로부터 다른 테이블 이름을 요청합니다. 이 메서드는 모든 페이지를 검색할 때까지 호출을 반복합니다.

```
        // 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() {
            }
        });
```

GitHub의 전체 예제를 참조하세요.

`Consumer` 사용

다음 예제는 `SdkPublisher`의 `subscribe` 메서드를 사용해 `Consumer`로 각 항목을 처리합니다.

```
        // Use a Consumer.
        CompletableFuture<Void> future = publisher.subscribe(System.out::println);
        future.get();
```

GitHub의 전체 예제를 참조하세요.

타사 라이브러리 사용

사용자 지정 구독자를 구현하는 대신 타사 라이브러리를 사용할 수 있습니다. 이 예제는 RxJava의 사용을 보여 주지만 반응형 스트림 인터페이스를 구현하는 모든 라이브러리를 사용할 수 있습니다. 해당 라이브러리에 대한 자세한 내용은 GitHub의 RxJava 위키 페이지를 참조하세요.

라이브러리를 사용하려면 종속성으로 추가합니다. 이 예에서는 Maven을 사용하는 경우에 사용할 POM 조각을 알려줍니다.

POM 항목

```
<dependency>
      <groupId>io.reactivex.rxjava3</groupId>
      <artifactId>rxjava</artifactId>
      <version>3.1.6</version>
</dependency>
```

코드

```
        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);
```

GitHub의 전체 예제를 참조하세요.

AWS SDK for Java 2.x에서 웨이터 사용

AWS SDK for Java 2.x의 웨이터 유틸리티를 사용하면 리소스에 대한 작업을 수행하기 전에 AWS 리소스가 지정된 상태에 있는지 확인할 수 있습니다.

웨이터는 원하는 상태에 도달할 때까지(또는 AWS 리소스가 원하는 상태에 도달하지 못할 것이라는 결정이 내려질 때까지) DynamoDB 테이블이나 Amazon S3 버킷과 같은 리소스를 폴링하는 데 사용되는 추상화입니다. 번거롭고 오류가 발생하기 쉬운 AWS 리소스를 지속적으로 폴링하는 로직을 작성하는 대신, 웨이터를 사용하여 리소스를 폴링하고 리소스가 준비된 후에도 코드가 계속 실행되도록 할 수 있습니다.

사전 조건

프로젝트에서 AWS SDK for Java와 함께 웨이터를 사용하려면 먼저 AWS SDK for Java2.x 설정의 단계를 완료해야 합니다.

또한 AWS SDK for Java 버전 `2.15.0` 또는 그 이상의 버전을 사용하도록 프로젝트 종속성(예: `pom.xml` 또는 `build.gradle` 파일)을 구성해야 합니다.

예:

```
<project>
  <dependencyManagement>
   <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>2.27.21</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
   </dependencies>
  </dependencyManagement>
</project>
```

웨이터 사용

웨이터 객체를 인스턴스화하려면 먼저 서비스 클라이언트를 생성해야 합니다. 서비스 클라이언트의 `waiter()` 메서드를 웨이터 객체의 값으로 설정합니다. 웨이터 인스턴스가 존재하면 적절한 코드를 실행하도록 응답 옵션을 설정합니다.

동기 프로그래밍

다음 코드 조각은 DynamoDB 테이블이 존재하고 ACTIVE 상태가 될 때까지 기다리는 방법을 보여줍니다.

```
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);
```

비동기 프로그래밍

다음 코드 조각은 DynamoDB 테이블이 더 이상 존재하지 않을 때까지 기다리는 방법을 보여줍니다.

```
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();
```

웨이터 설정

빌더에서 `overrideConfiguration()`를 사용하여 웨이터 구성을 사용자 지정할 수 있습니다. 일부 작업의 경우 요청 시 사용자 지정 구성을 적용할 수 있습니다.

웨이터 구성

다음 코드 조각은 웨이터의 구성을 재정의하는 방법을 보여줍니다.

```
// 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();
```

특정 요청에 대한 구성 재정의

다음 코드 조각은 요청별로 웨이터의 구성을 재정의하는 방법을 보여줍니다. 단, 일부 작업에만 사용자 지정 가능한 구성이 있습니다.

```
waiter.waitUntilTableNotExists(b -> b.tableName("myTable"),
               o -> o.maxAttempts(10));

asyncWaiter.waitUntilTableExists(b -> b.tableName("myTable"),
                 o -> o.waitTimeout(Duration.ofMinutes(1)));
```

코드 예제

DynamoDB와 함께 웨이터를 사용하는 전체 예제를 보려면 AWS 코드 예제 리포지토리의 CreateTable.java 를 참조하세요.

Amazon S3와 함께 웨이터를 사용하는 전체 예제를 보려면 AWS 코드 예제 리포지토리의 S3BucketOps.java 를 참조하세요.

문제 해결 FAQ

애플리케이션에서 AWS SDK for Java 2.x를 사용할 때 이 주제에 나열된 런타임 오류가 발생할 수 있습니다. 여기에서 제안 사항을 사용하면 근본 원인을 파악하고 오류를 해결하는 데 도움이 됩니다.

'`java.net.SocketException`: 연결 재설정' 또는 '서버가 응답을 완료하지 못함' 오류 해결 방법

연결 재설정 오류는 호스트, AWS 서비스 또는 중개자(예: NAT 게이트웨이, 프록시, 로드 밸런서)가 요청이 완료되기 전에 연결을 종료했음을 나타냅니다. 원인에는 여러 가지가 있으므로 솔루션을 찾으려면 연결이 종료되는 이유를 알아야 합니다. 다음 항목으로 인해 일반적으로 연결이 종료됩니다.

- 연결이 비활성 상태입니다. 이는 스트리밍 작업에서 데이터가 일정 기간 동안 유선에 기록되지 않아 중개자가 연결이 멈춘 것으로 감지하여 종료되는 경우에 흔히 발생합니다. 이를 방지하려면 애플리케이션에서 데이터를 적극적으로 다운로드하거나 업로드해야 합니다.
- HTTP 또는 SDK 클라이언트를 종료했습니다. 사용 중인 리소스는 종료하지 않아야 합니다.
- 잘못 구성된 프록시입니다. 문제가 해결되었는지 확인하기 위해 구성한 프록시를 우회해 봅니다. 이렇게 해서 문제가 해결되면 어떤 이유로든 프록시는 연결을 종료합니다. 특정 프록시를 조사하여 연결을 닫는 이유를 확인합니다.

문제를 파악할 수 없는 경우 네트워크의 클라이언트 엣지에서 영향을 받는 연결에 대해 TCP 덤프를 실행합니다(예: 제어하는 프록시 이후).

AWS 엔드포인트가 TCP RST를 전송(재설정)하는 경우 영향을 받는 서비스에 문의하여 재설정이 발생하는 이유를 확인할 수 있는지 알아봅니다. 문제가 발생한 때의 요청 ID와 타임스탬프를 제공할 준비를 합니다. 또한 AWS 지원 팀은 애플리케이션이 전송 및 수신하는 바이트와 시기를 정확하게 보여주는 유선 로그의 이점을 누릴 수 있습니다.

'연결 제한 시간'을 수정하는 방법

연결 제한 시간 오류는 호스트, AWS 서비스 또는 중개자(예: NAT 게이트웨이, 프록시, 로드 밸런서)가 구성된 연결 제한 시간 내에 서버와 새 연결을 설정하지 못했음을 나타냅니다. 다음 항목에서는 이 문제의 일반적인 원인을 설명합니다.

- 구성된 연결 제한 시간이 너무 짧습니다. 기본적으로 연결 제한 시간은 AWS SDK for Java 2.x에서 2초입니다. 연결 제한 시간을 너무 작게 설정하면 이 오류가 발생할 수 있습니다. 권장 연결 제한 시간은 리전 내 호출만 수행하는 경우 1초, 크로스 리전 요청을 수행하는 경우 3초입니다.
- 잘못 구성된 프록시입니다. 문제 해결 여부를 보려면 구성한 프록시를 우회해 봅니다. 이렇게 해서 문제가 해결되면 연결 제한 시간의 원인은 프록시입니다. 특정 프록시를 조사하여 이 문제가 발생하는 이유를 확인합니다.

문제를 파악할 수 없는 경우 네트워크의 클라이언트 엣지에서 영향을 받는 연결에 대해 TCP 덤프를 실행하여(예: 제어하는 프록시 이후) 네트워크 문제를 조사합니다.

'`java.net.SocketTimeoutException`: 읽기 제한 시간'을 수정하는 방법

읽기 제한 시간 오류는 JVM이 기본 운영 체제에서 데이터를 읽으려고 시도했지만 SDK를 통해 구성된 시간 내에 데이터가 반환되지 않았음을 나타냅니다. 이 오류는 운영 체제, AWS 서비스 또는 중개자(예: NAT 게이트웨이, 프록시, 로드 밸런서)가 JVM에서 예상한 시간 내에 데이터를 전송하지 못하는 경우에 발생할 수 있습니다. 원인에는 여러 가지가 있으므로 솔루션을 찾으려면 데이터가 반환되지 않는 이유를 알아야 합니다.

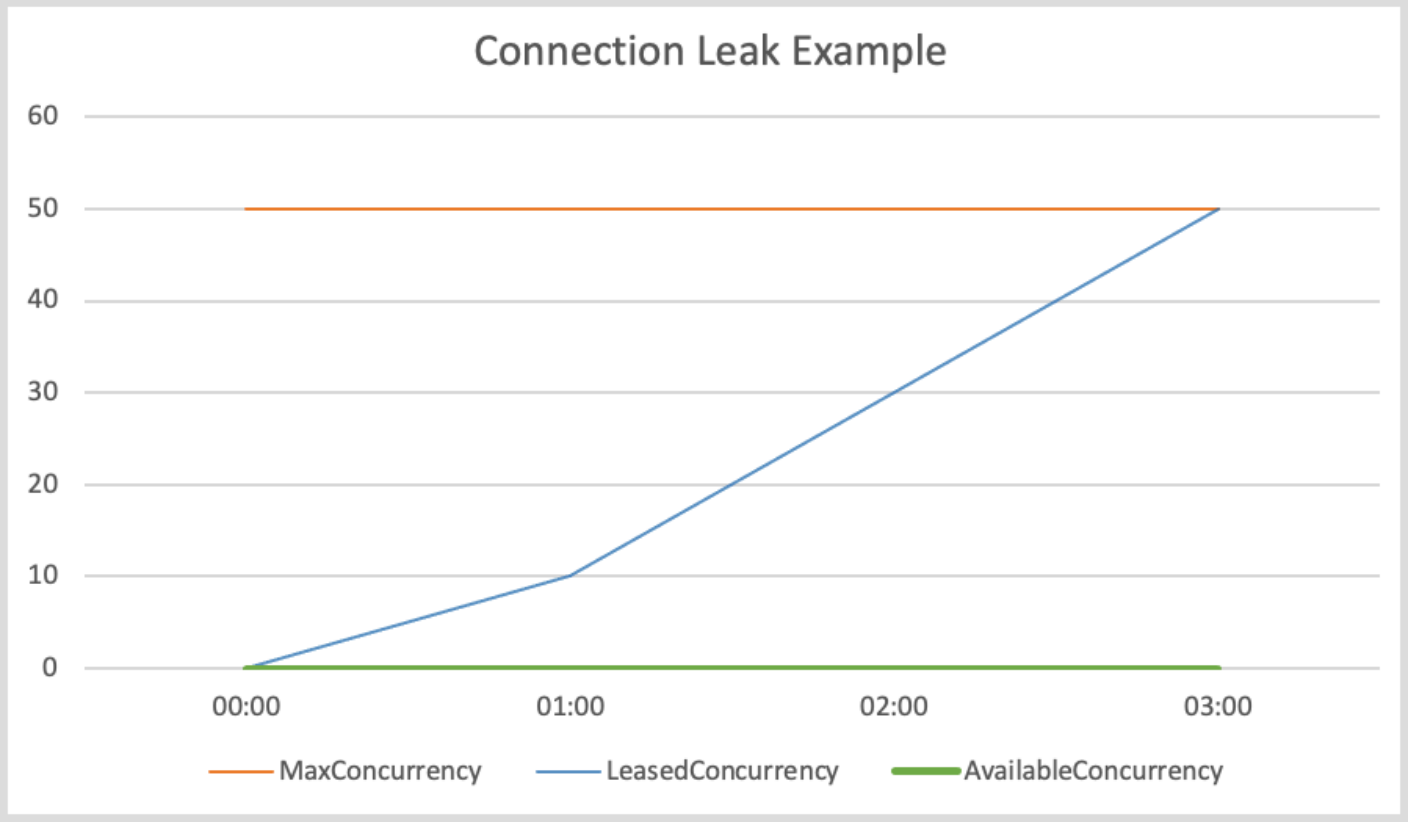
네트워크의 클라이언트 엣지에서 영향을 받는 연결에 대해 TCP 덤프를 실행해 봅니다(예: 제어하는 프록시 이후).

AWS 엔드포인트가 TCP RST(재설정)를 전송하고 있는 경우 영향을 받는 서비스에 문의하세요. 문제가 발생한 때의 요청 ID와 타임스탬프를 제공할 준비를 합니다. 또한 AWS 지원 팀은 애플리케이션이 전송 및 수신하는 바이트와 시기를 정확하게 보여주는 유선 로그의 이점을 누릴 수 있습니다.

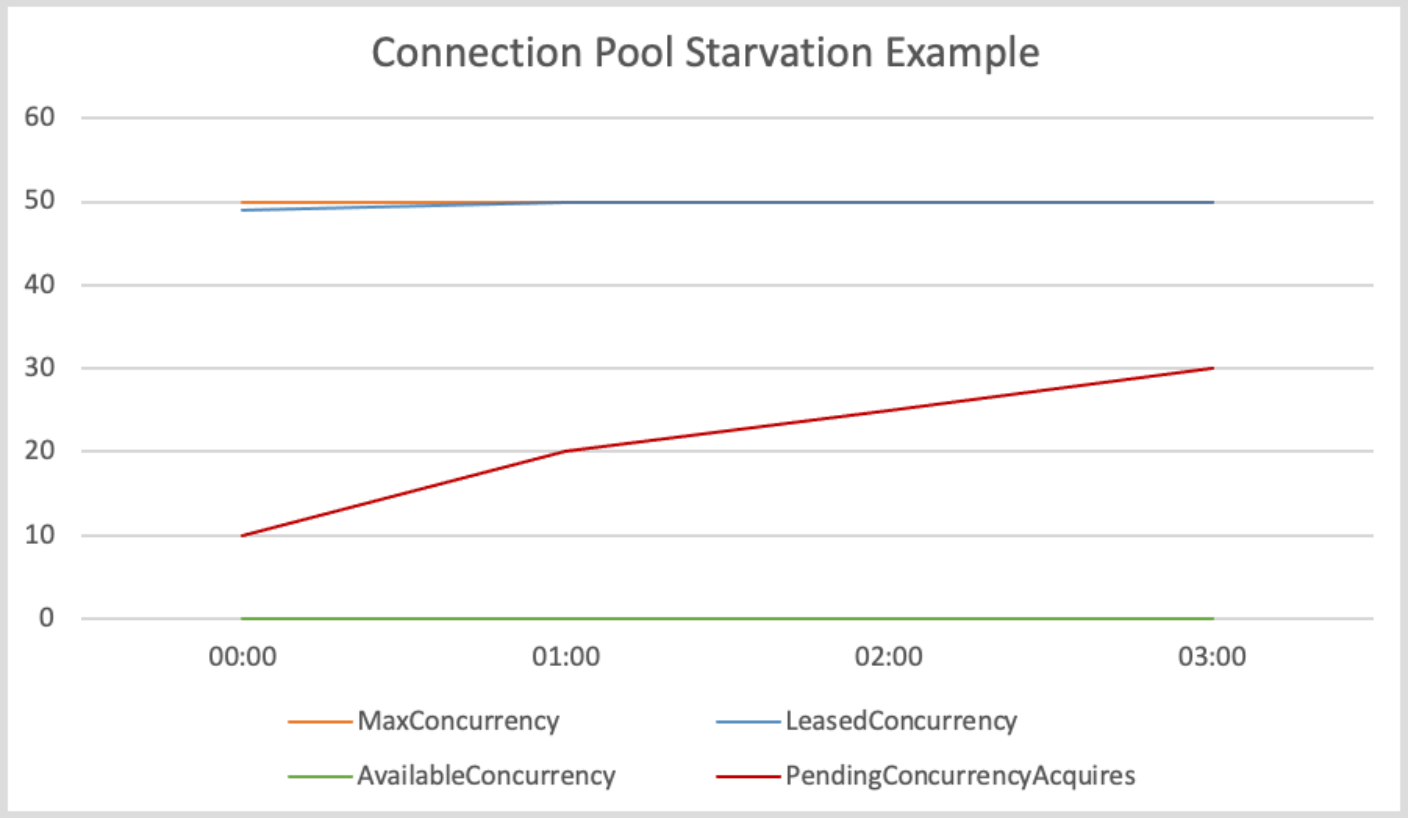
'HTTP 요청을 실행할 수 없음: 풀에서 연결을 기다리는 제한 시간' 오류를 해결하는 방법

이 오류는 요청이 지정된 최대 시간 내에 풀에서 연결을 가져올 수 없음을 나타냅니다. 문제를 해결하려면 SDK 클라이언트 측 지표를 사용하여 Amazon CloudWatch에 지표를 게시하는 것이 좋습니다. HTTP 지표는 근본 원인의 범위를 좁히는 데 도움이 될 수 있습니다. 다음 항목에서는 이 오류의 일반적인 원인을 설명합니다.

- 연결 누수입니다. `LeasedConcurrency` , `AvailableConcurrency` 및 `MaxConcurrency` 지표를 확인하여 이를 조사할 수 있습니다. `LeasedConcurrency`가 `MaxConcurrency`에 도달할 때까지 증가하지만 감소하지 않으면 연결 누수가 발생할 수 있습니다. 누수의 일반적인 원인은 S3 `getObject` 메서드 등의 스트리밍 작업이 종료되지 않기 때문입니다. 애플리케이션은 가능한 한 빨리 입력 스트림에서 모든 데이터를 읽고 나중에 입력 스트림을 종료하는 것이 좋습니다. 다음 차트는 연결 누수에 대한 SDK 지표를 보여줍니다.



- 연결 풀 결핍입니다. 이는 요청 속도가 너무 높고 구성된 연결 풀 크기가 요청 수요를 충족할 수 없는 경우에 발생할 수 있습니다. 기본 연결 풀 크기는 50이며 풀의 연결이 최대에 도달하면 HTTP 클라이언트는 연결을 사용할 수 있을 때까지 수신 요청을 대기열에 넣습니다. 다음 차트는 연결 풀 결핍에 대한 SDK 지표의 모습을 보여줍니다.



이 문제를 완화하려면 다음 작업 중 하나를 수행하는 것이 좋습니다.

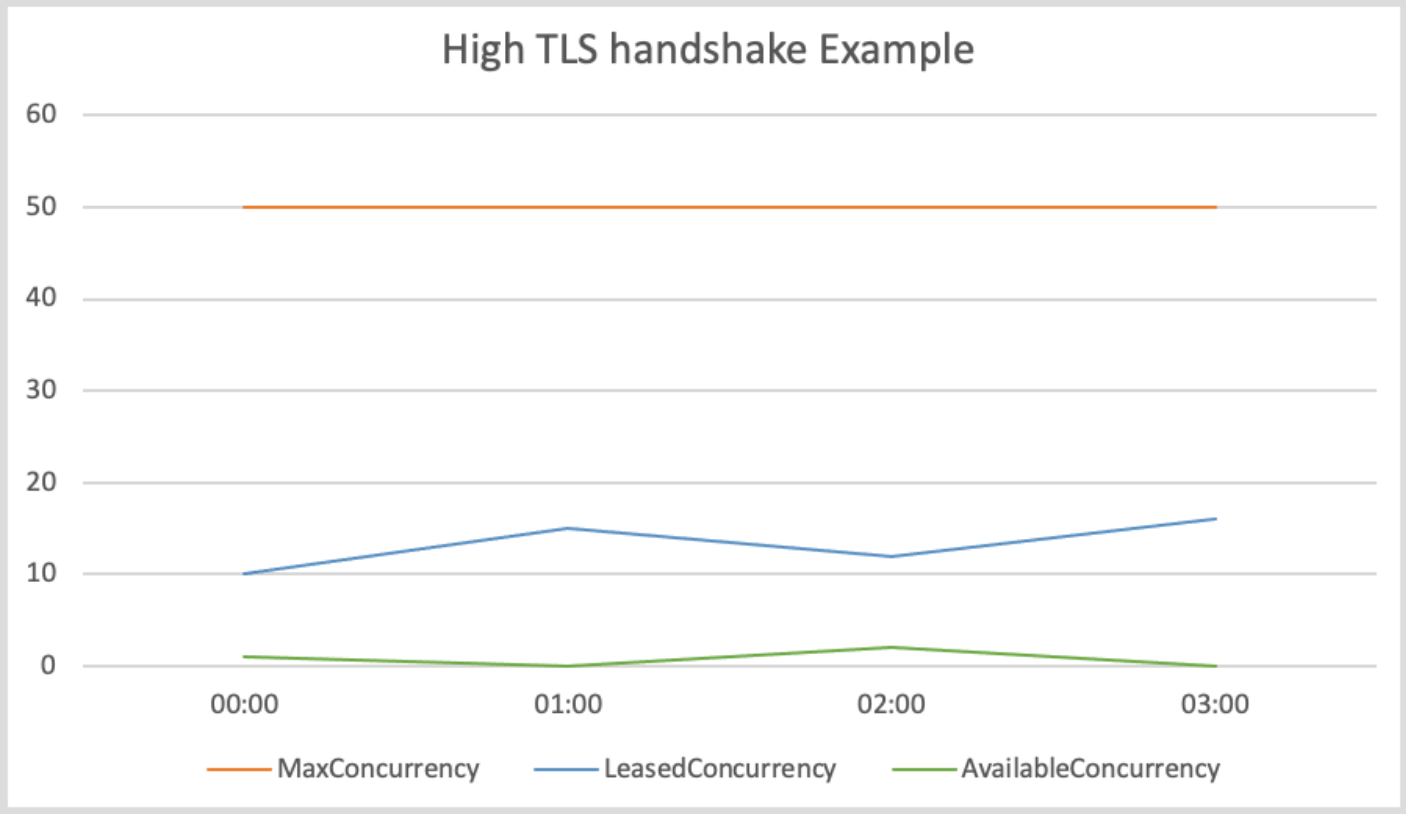
- 연결 풀 크기를 늘립니다.
- 획득 제한 시간을 늘립니다.
- 요청 속도를 늦춥니다.

최대 연결 수를 늘리면 클라이언트 처리량이 증가할 수 있습니다(네트워크 인터페이스가 이미 완전히 활용되지 않은 경우). 그러나 결국 프로세스에서 사용하는 파일 설명자 수에 대한 운영 체제 제한에 도달할 수 있습니다. 이미 네트워크 인터페이스를 완전히 사용하고 있거나 연결 수를 더 이상 늘릴 수 없는 경우 획득 제한 시간을 늘립니다. 값이 늘면 제한 시간을 초과하기 전에 요청을 통해 연결을 획득할 수 있는 추가 시간을 확보할 수 있습니다. 연결 시간이 확보되지 않으면 후속 요청 역시 제한 시간을 초과합니다.

처음 두 메커니즘을 사용하여 문제를 해결할 수 없는 경우 다음 옵션을 시도하여 요청 속도를 늦춥니다.

- 대규모 트래픽 버스트가 클라이언트에 과부하를 주지 않도록 요청을 완화합니다.
- AWS 서비스에 대한 호출을 더 효율적으로 수행할 수 있습니다.
- 요청을 보내는 호스트 수를 늘립니다.

- I/O 스레드가 사용 중입니다. 이는 NettyNioAsyncHttpClient에서 비동기식 SDK 클라이언트를 사용하는 경우에만 적용됩니다. 풀에서 연결을 사용할 수 있고 AvailableConcurrency 지표가 낮지 않지만 ConcurrencyAcquireDuration이 높으면 I/O 스레드가 요청을 처리할 수 없기 때문일 수 있습니다. 이는 I/O 스레드를 차단할 수 있으므로 Runnable:run을 향후 완료 실행자로 전달하고 향후 응답 완료 체인에서 시간이 많이 걸리는 작업을 수행하지 않아야 합니다. 그렇지 않은 경우 eventLoopGroupBuilder 메서드를 사용하여 I/O 스레드 수를 늘리는 것이 좋습니다. 참조용으로 NettyNioAsyncHttpClient 인스턴스의 기본 I/O 스레드 수는 호스트의 CPU 코어 수의 2배입니다.
- TLS 핸드셰이크 지연 시간이 높습니다. AvailableConcurrency 지표가 거의 0이고 LeasedConcurrency는 MaxConcurrency보다 낮은 경우 TLS 핸드셰이크 지연 시간이 높기 때문일 수 있습니다. 다음 차트는 높은 TLS 핸드셰이크 지연 시간에 대한 SDK 지표의 모습을 보여줍니다.



CRT를 기반으로 하지 않는 Java SDK에서 제공하는 HTTP 클라이언트의 경우 TLS 로그를 사용하여 TLS 문제를 해결해 봅니다. AWS CRT 기반 HTTP 클라이언트의 경우 AWS CRT 로그를 사용해 봅니다. AWS 엔드포인트가 TLS 핸드셰이크를 수행하는 데 시간이 오래 걸리는 것으로 보이면 영향을 받는 서비스에 문의해야 합니다.

NoClassDefFoundError, NoSuchMethodError 또는 NoSuchFieldError를 수정하는 방법

NoClassDefFoundError는 런타임에 클래스를 로드할 수 없음을 나타냅니다. 이 오류의 가장 일반적인 2가지 원인은 다음과 같습니다.

- JAR이 누락되었거나 잘못된 버전의 JAR이 클래스 경로에 있기 때문에 클래스가 클래스 경로에 존재하지 않습니다.
- 정적 이니셜라이저가 예외를 발생시켜 클래스를 로드하지 못했습니다.

마찬가지로 NoSuchMethodError 및 NoSuchFieldError는 일반적으로 JAR 버전이 일치하지 않아 발생합니다. 다음 단계를 수행하는 것을 권장합니다.

1. 종속성을 확인하여 모든 SDK jar에서 동일한 버전을 사용하고 있는지 확인합니다. 클래스, 메서드 또는 필드를 찾을 수 없는 가장 일반적인 이유는 새 클라이언트 버전으로 업그레이드하지만 이전 '공유' SDK 종속성 버전을 계속 사용하는 경우입니다. 새 클라이언트 버전은 최신 '공유' SDK 종속성에만 있는 클래스를 사용하려고 할 수 있습니다. mvn dependency:tree 또는 gradle dependencies(Gradle용)를 실행하여 SDK 라이브러리 버전이 모두 일치하는지 확인합니다. 향후 이 문제 발생을 완전히 방지하려면 BOM(Bill of Materials)을 사용하여 SDK 모듈 버전을 관리하는 것이 좋습니다.

   다음 예제에서는 혼합 SDK 버전의 예를 보여줍니다.

   ```
   [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
   ```

   dynamodb의 버전은 2.20.00이고 aws-core의 버전은 2.13.19입니다. aws-core 아티팩트 버전도 2.20.00이어야 합니다.
2. 로그 초기에 문을 확인하여 정적 초기화 실패로 인해 클래스가 로드되지 않는지 확인합니다. 클래스가 처음 로드되지 않으면 클래스를 로드할 수 없는 이유를 지정하는 더 유용한 다른 예외가 발생할 수 있습니다. 이 잠재적으로 유용한 예외는 한 번만 발생하므로 이후 로그 문은 클래스를 찾을 수 없음을 보고합니다.
3. 배포 프로세스를 확인하여 애플리케이션과 함께 필요한 JAR 파일을 실제로 배포하는지 확인합니다. 올바른 버전으로 구축하고 있지만 애플리케이션에 대한 클래스 경로를 만드는 프로세스가 필수 종속성을 제외하고 있는 경우일 수 있습니다.

'`SignatureDoesNotMatch`' 오류 또는 '계산한 요청 서명이 제공한 서명과 일치하지 않음' 오류를 해결하는 방법

`SignatureDoesNotMatch` 오류는 AWS SDK for Java에서 생성된 서명과 AWS 서비스에서 생성된 서명이 일치하지 않음을 나타냅니다. 다음 항목에서는 잠재적 원인을 설명합니다.

- 프록시 또는 중개자가 요청을 수정합니다. 예를 들어 프록시 또는 로드 밸런서는 SDK에서 서명한 헤더, 경로 또는 쿼리 문자열을 수정할 수 있습니다.
- 서비스 및 SDK는 각각에서 서명할 문자열을 생성할 때 요청을 인코딩하는 방식이 다릅니다.

이 문제를 디버깅하려면 SDK에 대한 디버그 로깅을 사용하는 것이 좋습니다. 오류를 재현하고 SDK가 생성한 정식 요청을 찾습니다. 로그에서 정식 요청에는 레이블이 `AWS4 String to sign: ...`으로 지정되고 서명할 문자열에는 `AWS4 Canonical Request: ...` 레이블이 지정됩니다.

예를 들어 프로덕션 환경에서만 재현할 수 있기 때문에 디버깅을 사용할 수 없는 경우 오류가 발생하면 요청에 대한 정보를 로깅하는 로직을 애플리케이션에 추가합니다. 그런 다음 해당 정보를 사용하여 디버그 로깅이 사용된 통합 테스트 시 프로덕션 외부에서 오류를 복제할 수 있습니다.

서명할 정식 요청과 문자열을 수집한 후 AWS 서명 버전 4 사양과 비교하여 SDK가 서명할 문자열을 생성한 방식에 문제가 있는지 확인합니다. 문제가 있는 것 같으면 AWS SDK for Java에 대한 GitHub 버그 보고서를 만들 수 있습니다.

문제가 전혀 표시되지 않으면 서명할 SDK의 문자열을 문자열과 비교하여 일부 AWS 서비스가 실패 응답의 일부로 반환된다고 서명할 수 있습니다(예: Amazon S3). 사용할 수 없는 경우 영향을 받는 서비스에 문의하여 비교를 위해 생성한 정식 요청과 문자열을 확인해야 합니다. 이러한 비교는 서비스와 클라이언트 간의 요청 또는 인코딩 차이를 수정했을 수 있는 중개자를 식별하는 데 도움이 될 수 있습니다.

요청 서명에 대한 자세한 배경 정보는 AWS Identity and Access Management 사용 설명서의 AWS API 요청에 서명을 참조하세요.

Example표준 요청

```
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
```

Example서명할 문자열

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

'**java.lang.IllegalStateException**: 연결 풀 종료' 오류를 해결하는 방법

이 오류는 기본 Apache HTTP 연결 풀이 종료되었음을 나타냅니다. 다음 항목에서는 잠재적 원인을 설명합니다.

- SDK 클라이언트가 조기에 종료되었습니다. SDK는 연결된 클라이언트가 종료될 때만 연결 풀을 종료합니다. 사용 중인 리소스는 종료하지 않아야 합니다.
- **java.lang.Error**가 발생했습니다. `OutOfMemoryError` 등의 오류로 인해 Apache HTTP 연결 풀이 종료됩니다. 로그에서 오류 스택 추적을 검사합니다. 또한 코드를 검토하여 `Throwable` 또는 `Error`를 포착하지만 표시에서 오류를 방지하는 출력을 포함하는 위치를 찾습니다. 코드가 오류를 보고하지 않는 경우 정보가 로그되도록 코드를 다시 작성합니다. 로그된 정보는 오류의 근본 원인을 확인하는 데 도움이 됩니다.
- 종료된 후 **DefaultCredentialsProvider#create()**에서 반환된 자격 증명 공급자를 사용하려고 했습니다. `DefaultCredentialsProvider#create`는 싱글톤 인스턴스를 반환하므로 종료되어 있고 코드가 `resolveCredentials` 메서드를 호출하면 캐시된 자격 증명(또는 토큰)이 만료된 후 예외가 발생합니다.

  다음 예제와 같이 `DefaultCredentialsProvider`가 종료된 위치가 있는지 코드를 확인합니다.

  - 싱글톤 인스턴스는 `DefaultCredentialsProvider#close().`를 호출하여 종료됩니다.

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

```
// Make calls to AWS ###.

defaultCredentialsProvider.close();  // Explicit close.

// Make calls to AWS ###.

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

- try-with-resources 블록에서 `DefaultCredentialsProvider#create()`를 호출합니다.

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

    // Make calls to AWS ###.

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

// Make calls to AWS ###.

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

코드가 싱글톤 인스턴스를 종료하고 `DefaultCredentialsProvider`를 사용하여 자격 증명을 해결해야 하는 경우 `DefaultCredentialsProvider.builder().build()`를 호출하여 새로운 비싱글톤 인스턴스를 만듭니다.

‘AwsCredentialsProviderChain 체인의 공급자로부터 자격 증명을 로드할 수 없음’ 오류를 수정하는 방법

이 오류는 AWS SDK for Java 2.x가 기본 자격 증명 공급자 체인의 자격 증명 공급자를 통해 유효한 AWS 자격 증명을 찾을 수 없음을 나타냅니다. SDK는 특정 순서로 자격 증명을 자동으로 검색하며, 이 오류는 체인의 모든 공급자가 유효한 자격 증명을 제공하지 못할 때 발생합니다.

전체 오류 메시지는 일반적으로 다음과 같습니다(가독성을 높이기 위해 줄 끝 및 들여쓰기가 추가됨).

```
Unable to load credentials from any of the providers in the chain
 AwsCredentialsProviderChain(
    credentialsProviders=[
        SystemPropertyCredentialsProvider(),
        EnvironmentVariableCredentialsProvider(),
        WebIdentityTokenCredentialsProvider(),
        ProfileCredentialsProvider(profileName=default,
 profileFile=ProfileFile(sections=[])),
        ContainerCredentialsProvider(),
        InstanceProfileCredentialsProvider()
    ]) : [
        SystemPropertyCredentialsProvider(): Unable to load credentials from system
 settings.
        Access key must be specified either via environment variable
 (AWS_ACCESS_KEY_ID)
        or system property (aws.accessKeyId).,

        EnvironmentVariableCredentialsProvider(): Unable to load credentials from
 system settings.
        Access key must be specified either via environment variable
 (AWS_ACCESS_KEY_ID)
        or system property (aws.accessKeyId).,

        WebIdentityTokenCredentialsProvider(): To use web identity tokens, the 'sts'
 service module
        must be on the class path.,

        ProfileCredentialsProvider(profileName=default,
 profileFile=ProfileFile(sections=[])):
        Profile file contained no credentials for profile 'default':
 ProfileFile(sections=[]),

        ContainerCredentialsProvider(): Cannot fetch credentials from container -
 neither
        AWS_CONTAINER_CREDENTIALS_FULL_URI or AWS_CONTAINER_CREDENTIALS_RELATIVE_URI
 environment
        variables are set.,

        InstanceProfileCredentialsProvider(): Failed to load credentials from IMDS.]
```

일반적인 원인 및 솔루션

자격 증명 구성 검토

기본 자격 증명 공급자를 사용하는 경우(자격 증명을 명시적으로 구성하지 않고 `ServiceClient.create()` 호출) SDK는 특정 순서로 자격 증명을 검색합니다. 기본 자격 증명 공급자 체인이 작동하는 방식을 검토하여 SDK가 확인하는 자격 증명과 순서를 파악합니다.

사용하려는 자격 증명 구성 메서드가 환경에 올바르게 설정되어 있는지 확인합니다.

Amazon EC2 인스턴스의 경우

- IAM 역할 확인: IAM 역할이 인스턴스에 연결되어 있는지 확인합니다.
- 간헐적 IMDS 장애: 간헐적 장애(일반적으로 수백 밀리초 지속)가 발생하는 경우 이는 일반적으로 인스턴스 메타데이터 서비스(IMDS)에 도달하는 일시적인 네트워크 문제를 나타냅니다.

  솔루션

  - 디버그 로깅을 사용하여 장애의 타이밍 및 빈도 분석
  - 자격 증명 관련 장애에 대해 애플리케이션에서 재시도 로직 구현 고려
  - 인스턴스와 IMDS 엔드포인트 간의 네트워크 연결 문제 확인

컨테이너 환경의 경우

작업 역할(Amazon ECS) 또는 서비스 계정(Amazon EKS)이 구성되어 있고 필요한 환경 변수가 설정되어 있는지 확인합니다.

로컬 개발의 경우

환경 변수, 자격 증명 파일 또는 IAM Identity Center 구성이 있는지 확인합니다.

웹 ID 페더레이션의 경우

- 구성 확인: 웹 자격 증명 토큰 파일이 존재하고 필요한 환경 변수가 구성되어 있는지 확인합니다.
- STS 모듈 종속성 누락: 오류(`To use web identity tokens, the 'sts' service module must be on the class path`)가 표시되면 STS 모듈을 종속성으로 추가해야 합니다. 이는 Amazon EKS Pod Identity 또는 기타 웹 자격 증명 토큰 인증을 사용할 때 흔히 발생합니다.

  솔루션: 프로젝트 종속성에 STS 모듈을 추가합니다.

  - ```
    <dependency>
    ```
    ```

```
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>sts</artifactId>
</dependency>
```

일부 서비스의 경우 `aws-query-protocol` 종속성이 필요할 수 있습니다.

```
<dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>aws-query-protocol</artifactId>
</dependency>
```

네트워크 또는 프록시 연결 문제

자격 증명 공급자 체인에 `Connection refused` 오류가 표시되는 경우 이는 일반적으로 SDK가 AWS 엔드포인트에 도달하려고 할 때의 네트워크 연결 문제를 나타냅니다.

솔루션

- 프록시 서버를 사용하는 경우 프록시 구성 확인
- 네트워크에서 AWS 엔드포인트에 대한 아웃바운드 HTTPS 연결을 허용하는지 확인
- 디버그 로깅을 사용하여 자세한 연결 시도 확인
- `curl` 등의 도구를 사용하여 AWS 엔드포인트에 대한 네트워크 액세스를 확인하는 연결 테스트

AWS Lambda를 위한 SDK 시작 시간 단축

AWS SDK for Java 2.x 의 목표 중 하나는 AWS Lambda 함수의 시작 지연 시간을 줄이는 것입니다. SDK에는 시작 시간을 줄이는 변경 사항이 포함되어 있으며, 이에 대해서는 이 항목의 마지막 부분에서 설명합니다.

먼저, 이 항목에서는 콜드 스타트 시간을 줄이기 위해 적용할 수 있는 변경 사항에 초점을 맞춥니다. 여기에는 코드 구조 및 서비스 클라이언트 구성 변경이 포함됩니다.

AWS CRT 기반 HTTP 클라이언트 사용

AWS Lambda로 작업하려면 동기식 시나리오의 경우 `AwsCrtHttpClient`, 비동기식 시나리오의 경우 `AwsCrtAsyncHttpClient`를 사용하는 것이 좋습니다.

이 가이드의 the section called “AWS CRT 기반 HTTP 클라이언트 구성” 주제에서는 HTTP 클라이언트 사용 시 이점, 종속성을 추가하는 방법, 서비스 클라이언트에서 이를 사용하도록 구성하는 방법을 설명합니다.

사용하지 않는 HTTP 클라이언트 종속성을 제거

AWS CRT 기반 클라이언트를 명시적으로 사용하는 것과 함께 SDK에서 기본적으로 제공하는 다른 HTTP 클라이언트를 제거할 수 있습니다. 로드해야 하는 라이브러리 수가 적을수록 Lambda 시작 시간이 단축되므로 JVM이 로드해야 하는 사용되지 않은 아티팩트를 제거해야 합니다.

Maven `pom.xml` 파일의 다음 코드 조각은 Apache 기반 HTTP 클라이언트와 Netty 기반 HTTP 클라이언트의 제외를 보여줍니다. (이 클라이언트는 AWS CRT 기반 클라이언트를 사용할 때 필요하지 않습니다.) 이 예제에서는 S3 클라이언트 종속성에서 HTTP 클라이언트 아티팩트를 제외하고 AWS CRT 기반 HTTP 클라이언트에 대한 액세스를 허용하는 `aws-crt-client` 아티팩트를 추가합니다.

```
<project>
    <properties>
        <aws.java.sdk.version>2.27.21</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**
>
> pom.xml 파일의 모든 서비스 클라이언트 종속성에 <exclusions> 요소를 추가합니다.

바로가기 검색이 가능하도록 서비스 클라이언트를 구성

리전 지정

서비스 클라이언트를 생성할 때는 서비스 클라이언트 빌더에서 region 메서드를 호출하세요. 이렇게 하면 여러 곳에서 AWS 리전 정보를 확인하는 SDK의 기본 리전 조회 프로세스가 단축됩니다.

Lambda 코드를 리전과 독립적으로 유지하려면 region 메서드 내에서 다음 코드를 사용하세요. 이 코드는 Lambda 컨테이너에서 설정한 AWS_REGION 환경 변수에 액세스합니다.

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

EnvironmentVariableCredentialProvider 사용

리전 정보에 대한 기본 조회 동작과 마찬가지로 SDK는 여러 위치에서 자격 증명을 찾습니다. 서비스 클라이언트를 구축할 때 EnvironmentVariableCredentialProvider를 지정하면 자격 증명용 SDK의 조회 프로세스에서 시간을 절약할 수 있습니다.

> **Note**
>
> 이 자격 증명 공급자를 사용하면 Lambda 함수에서 코드를 사용할 수 있지만, Amazon EC2 또는 기타 시스템에서는 이 기능이 작동하지 않을 수 있습니다.
> 어느 시점에서 Lambda SnapStart for Java를 사용하려는 경우 기
> 본 자격 증명 공급자 체인을 사용하여 자격 증명을 조회해야 합니다.

`EnvironmentVariableCredentialsProvider`를 지정하면 초기 자격 증명 조회가 작동하지만 SnapStart가 활성화되면 Java 런타임이 컨테이너 자격 증명 환경 변수를 설정합니다. 활성화 시 `EnvironmentVariableCredentialsProvider`(액세스 키 환경 변수)에서 사용하는 환경 변수는 Java SDK에서 사용할 수 없습니다.

다음 코드 조각은 Lambda 환경에서 사용하도록 적절하게 구성된 S3 서비스 클라이언트를 보여줍니다.

```
S3Client s3Client = S3Client.builder()

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

Lambda 함수 핸들러 외부에서 SDK 클라이언트 초기화

Lambda 핸들러 메서드 외부에서 SDK 클라이언트를 초기화하는 것이 좋습니다. 이렇게 하면 실행 컨텍스트를 재사용하는 경우 서비스 클라이언트의 초기화를 건너뛸 수 있습니다. 클라이언트 인스턴스와 해당 연결을 재사용하면 핸들러 메서드의 후속 호출이 더 빠르게 발생합니다.

다음 예제에서는 정적 팩토리 메서드를 사용하여 생성자에서 `S3Client` 인스턴스를 초기화합니다. Lambda 환경에서 관리하는 컨테이너를 재사용하는 경우 초기화된 `S3Client` 인스턴스가 재사용됩니다.

```
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.
    }
}
```

종속성 주입을 최소화

종속성 주입(DI) 프레임워크는 설정 프로세스를 완료하는 데 시간이 더 걸릴 수 있습니다. 또한 추가 종속성이 필요할 수 있으며, 이 경우 로드하는 데 시간이 걸립니다.

DI 프레임워크가 필요한 경우 Dagger와 같은 가벼운 DI 프레임워크를 사용하는 것이 좋습니다.

AWS Lambda를 대상으로 하는 Maven 아키타입 사용

AWS Java SDK 팀은 시작 시간을 최소화하면서 Lambda 프로젝트를 부트스트랩할 수 있는 Maven Archetype 템플릿을 개발했습니다. 아키타입에서 Maven 프로젝트를 구축하고 종속성이 Lambda 환경에 적합하게 구성되어 있는지 알 수 있습니다.

아키타입에 대해 자세히 알아보고 예제 배포를 진행하려면 이 블로그 게시물을 참조하세요.

Java용 Lambda SnapStart를 고려

런타임 요구 사항이 호환되는 경우 AWS에서 Java용 Lambda SnapStart를 제공합니다. Lambda SnapStart는 Java 기능의 시작 성능을 향상시키는 인프라 기반 솔루션입니다. 새 버전의 함수를 게시하면 Lambda SnapStart가 이를 초기화하고 메모리 및 디스크 상태에 대한 변경이 불가능하고 암호화된 스냅샷을 생성합니다. 그런 다음 SnapStart는 재사용을 위해 스냅샷을 캐싱합니다.

시작 시간에 영향을 미치는 버전 2.x 변경 사항

코드 변경 사항 외에도 SDK for Java 버전 2.x에는 시작 시간을 줄이는 세 가지 주요 변경 사항이 포함되어 있습니다.

- 초기화 시간을 개선하는 직렬화 라이브러리인 jackson-jr을 사용
- JDK의 일부인 날짜 및 시간 객체에 대해 java.time 라이브러리를 사용
- 로깅 facade에 Slf4j를 사용

추가 리소스

AWS Lambda 개발자 안내서에는 Java에만 국한되지 않는 Lambda 함수 개발을 위한 모범 사례에 대한 단원이 포함되어 있습니다.

AWS Lambda를 사용하는 Java로 클라우드 네이티브 애플리케이션을 구축하는 예제는 이 워크숍 콘텐츠를 참조하세요. 워크숍에서는 성능 최적화 및 기타 모범 사례에 대해 논의합니다.

시작 대기 시간을 줄이기 위해 미리 컴파일된 정적 이미지를 사용하는 것을 고려할 수 있습니다. 예를 들어 Java 2.x 및 Maven용 SDK를 사용하여 GraalVM 네이티브 이미지를 빌드할 수 있습니다.

AWS SDK for Java 2.x에서 `ContentStreamProvider` 구현

`ContentStreamProvider`는 AWS SDK for Java 2.x에서 입력 데이터의 여러 읽기를 허용하는 데 사용되는 추상적 방식입니다. 이 주제에서는 애플리케이션에서 `ContentStreamProvider`를 올바르게 구현하는 방법을 설명합니다.

SDK for Java 2.x는 전체 스트림을 읽어야 할 때마다 `ContentStreamProvider#newStream()` 메서드를 사용합니다. 전체 스트림에서 작동하려면 반환된 스트림이 항상 콘텐츠의 시작 부분에 있어야 하며 동일한 데이터를 포함해야 합니다.

다음 섹션에서는 이 동작을 올바르게 구현하는 방법에 대한 3가지 접근 방식을 제공합니다.

`mark()` 및 `reset()` 사용

아래 예제에서는 읽기를 시작하기 전에 생성자에서 `mark(int)`를 사용하여 스트림을 다시 시작으로 재설정할 수 있도록 합니다. `newStream()`을 호출할 때마다 스트림이 재설정됩니다.

```
public class MyContentStreamProvider implements ContentStreamProvider {
    private InputStream contentStream;

    public MyContentStreamProvider(InputStream contentStream) {
        this.contentStream = contentStream;
        this.contentStream.mark(MAX_LEN);
    }

    @Override
    public InputStream newStream() {
        contentStream.reset();
        return contentStream;
    }
}
```

`mark()` 및 `reset()`을 사용할 수 없는 경우 버퍼링 사용

스트림이 `mark()` 및 `reset()`을 직접 지원하지 않는 경우 먼저 스트림을 `BufferedInputStream`에 래핑하여 이전에 표시된 솔루션을 계속 사용할 수 있습니다.

```
public class MyContentStreamProvider implements ContentStreamProvider {
    private BufferedReader contentStream;

    public MyContentStreamProvider(InputStream contentStream) {
        this.contentStream = new BufferedInputStream(contentStream);
        this.contentStream.mark(MAX_LEN);
    }

    @Override
    public InputStream newStream() {
        contentStream.reset();
        return contentStream;
    }
}
```

새 스트림 만들기

더 간단한 접근 방식은 각 호출에서 데이터에 대한 새 스트림을 얻고 이전 스트림을 종료하는 것입니다.

```
public class MyContentStreamProvider implements ContentStreamProvider {
    private InputStream contentStream;

    @Override
    public InputStream newStream() {
        if (contentStream != null) {
            contentStream.close();
        }
        contentStream = openStream();
        return contentStream;
    }
}
```

DNS 이름 조회를 위한 JVM TTL 설정

Java 가상 머신(JVM)은 DNS 이름 조회를 캐시합니다. JVM은 호스트 이름을 IP 주소로 확인하는 경우 Time-To-Live(TTL)라고 하는 지정된 기간 동안 IP 주소를 캐시합니다.

AWS 리소스는 간헐적으로 변경되는 DNS 이름 항목을 사용하므로 TTL 값을 5초로 하여 JVM을 구성하는 것이 좋습니다. 이렇게 하면 리소스의 IP 주소가 변경될 때 애플리케이션이 DNS를 다시 쿼리하여 리소스의 새 IP 주소를 수신하고 사용할 수 있습니다.

일부 Java 구성에서는 JVM이 다시 시작될 때까지 DNS 항목을 새로 고치지 않도록 JVM 기본 TTL이 설정되기도 합니다. 따라서 애플리케이션이 실행 중인 동안 AWS 리소스의 IP 주소가 변경되는 경우 JVM을 수동으로 다시 시작하여 캐시된 IP 정보가 새로 고쳐질 때까지는 해당 리소스를 사용할 수 없게 됩니다. 이 경우 캐시된 IP 정보를 정기적으로 새로 고치도록 JVM의 TTL을 설정해야 합니다.

JVM TTL을 설정하는 방법

JVM의 TTL을 수정하려면 networkaddress.cache.ttl 보안 속성 값을 설정하고 Java 8의 경우 `$JAVA_HOME/jre/lib/security/java.security` 파일에서, Java 11 이상의 경우 `$JAVA_HOME/conf/security/java.security` 파일에서 `networkaddress.cache.ttl` 속성을 설정합니다.

다음은 5초로 설정된 TTL 캐시를 보여주는 `java.security` 파일의 코드 조각입니다.

```
#
# 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
...
```

`$JAVA_HOME` 환경 변수로 표시되는 JVM에서 실행되는 모든 애플리케이션은 이 설정을 사용합니다.

AWS SDK for Java에서 HTTP/2 작업

HTTP/2는 HTTP 프로토콜의 주요 내용 개정입니다. 이 새 버전에는 성능 개선을 위한 여러 기능 향상이 있습니다.

- 이진 데이터 인코딩으로 더욱 효율적인 데이터 전송을 제공합니다.
- 헤더 압축으로 클라이언트가 다운로드하는 오버헤드 바이트를 줄여 클라이언트로 더욱 빠르게 콘텐츠를 가져오는 데 도움이 됩니다. 이는 특히 이미 대역폭 제약이 있는 모바일 클라이언트에 유용합니다.

- 양방향 비동기 통신(멀티플렉싱)을 통해 클라이언트와 AWS 간에 여러 요청 및 응답 메시지가 여러 개의 연결 대신 단일 연결로 동시에 전송되어 성능이 개선됩니다.

최신 SDK로 업그레이드한 개발자는 사용하는 서비스에서 지원할 때 HTTP/2를 자동으로 사용합니다. 새로운 프로그래밍 인터페이스는 HTTP/2 기능을 원활하게 활용하게 애플리케이션을 빌드하는 새로운 방법을 제공합니다.

AWS SDK for Java 2.x에는 HTTP/2 프로토콜을 구현하는 이벤트 스트리밍에 대한 새 API가 있습니다. 이러한 새 API를 사용하는 방법에 대한 예는 Kinesis로 작업을 참조하세요.

AWS SDK for Java 2.x에서 AWS 서비스 호출

이 섹션에서는 간단한 자습서와 일부 AWS 서비스와의 작업 방법에 대한 가이드를 제공합니다. 전체 예제 세트는 코드 예제 섹션을 참조하세요.

주제

작업 CloudWatch

이 섹션에서는 AWS SDK for Java 2.x를 사용하여 Amazon CloudWatch를 프로그래밍하는 예제를 제공합니다.

Amazon CloudWatch 는 Amazon Web Services (AWS) 리소스와 AWS 실행 중인 애플리케이션을 실시간으로 모니터링합니다. CloudWatch 를 사용하여 리소스 및 애플리케이션에 대해 측정할 수 있는 변수인 지표를 수집하고 추적할 수 있습니다. CloudWatch 경보는 사용자가 정의한 규칙에 따라 모니터링 중인 리소스를 자동으로 변경하거나 알림을 보냅니다.

다음 예제에는 각 기술을 보여주는 데 필요한 코드만 포함되어 있습니다. 전체 예제 코드는 GitHub에 있습니다. 이 위치에서 단일 소스 파일을 다운로드하거나 리포지토리를 로컬로 복사하여 모든 예제를 빌드하고 실행할 수 있습니다.

주제

- 에서 지표 가져오기 CloudWatch
- 사용자 지정 지표 데이터를 CloudWatch에 게시
- CloudWatch 경보 작업
- Amazon CloudWatch Events 사용

에서 지표 가져오기 CloudWatch

지표 나열

CloudWatch 지표를 나열하려면 ListMetricsRequest를 생성하고 CloudWatchClient의 `listMetrics` 메서드를 호출합니다. `ListMetricsRequest`를 사용하여 반환된 지표를 네임스페이스, 지표 이름 또는 차원을 기준으로 필터링할 수 있습니다.

> **Note**
>
> AWS 서비스에서 게시하는 지표 및 차원 목록은 Amazon CloudWatch 사용 설명서의 Amazon CloudWatch 지표 및 차원 참조에서 확인할 수 있습니다.

가져오기

```
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;
```

코드

```
    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);
        }
    }
```

지표는 해당 `getMetrics` 메서드를 호출하여 ListMetricsResponse에 반환됩니다.

결과를 페이징할 수 있습니다. 다음 결과 배치를 검색하려면 응답 객체에서 `nextToken`를 호출하고 토큰 값을 사용하여 새 요청 객체를 빌드합니다. 그런 다음 새 요청을 사용해 다시 `listMetrics` 메서드를 호출합니다.

GitHub의 전체 예제를 참조하세요.

추가 정보

- Amazon CloudWatch API 참조의 ListMetrics

사용자 지정 지표 데이터를 CloudWatch에 게시

여러 AWS 서비스가 "AWS"로 시작하는 네임스페이스에 자체 지표를 게시합니다. 자체 네임스페이스를 사용하여 사용자 지정 지표 데이터를 게시할 수도 있습니다("AWS"로 시작하지 않는 한).

사용자 지정 지표 데이터 게시

자체 지표 데이터를 게시하려면 PutMetricDataRequest를 사용하여 CloudWatchClient의 `putMetricData` 메서드를 호출하세요. `PutMetricDataRequest`는 데이터에 사용할 사용자 지정 네임스페이스와, MetricDatum 객체의 데이터 포인트 자체에 대한 정보를 포함해야 합니다.

> **Note**
>
> "AWS"로 시작하는 네임스페이스는 지정할 수 없습니다. "AWS"로 시작하는 네임스페이스는 Amazon Web Services 제품용으로 예약되어 있습니다.

가져오기

```
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;
```

코드

```
    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);
    }
```

GitHub의 전체 예제를 참조하세요.

추가 정보

- Amazon CloudWatch 사용 설명서의 Amazon CloudWatch 지표를 사용합니다.
- Amazon CloudWatch 사용 설명서의 AWS 네임스페이스.
- Amazon CloudWatch API 참조의 PutMetricData.

CloudWatch 경보 작업

경보 만들기

CloudWatch 지표를 기반으로 경보를 생성하려면 경보 조건으로 채워진 PutMetricAlarmRequest를 사용하여 CloudWatchClient의 `putMetricAlarm` 메서드를 호출합니다.

가져오기

```
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;
```

코드

```
   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);
        }
    }
```

GitHub의 전체 예제를 참조하세요.

경보 나열

생성한 CloudWatch 경보를 나열하려면 결과에 대한 옵션을 설정하는 데 사용할 수 있는 DescribeAlarmsRequest를 사용하여 CloudWatchClient의 `describeAlarms` 메서드를 호출합니다.

가져오기

```
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;
```

코드

```
    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");
    }
```

`describeAlarms`에 의해 반환되는 DescribeAlarmsResponse에 대해 `MetricAlarms`를 호출하여 경보 목록을 가져올 수 있습니다.

결과를 페이징할 수 있습니다. 다음 결과 배치를 검색하려면 응답 객체에서 `nextToken`를 호출하고 토큰 값을 사용하여 새 요청 객체를 빌드합니다. 그런 다음 새 요청을 사용해 다시 `describeAlarms` 메서드를 호출합니다.

> **Note**
>
> CloudWatchClient의 `describeAlarmsForMetric` 메서드를 사용하여 특정 지표의 경보를 검색할 수도 있습니다. 이 메서드의 용도는 `describeAlarms`와 비슷합니다.

GitHub의 전체 예제를 참조하세요.

경보 삭제

CloudWatch 경보를 삭제하려면 삭제하려는 하나 이상의 경보 이름이 포함된 DeleteAlarmsRequest를 사용하여 CloudWatchClient의 `deleteAlarms` 메서드를 호출합니다.

가져오기

```
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;
```

코드

```
    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);
        }
    }
```

GitHub의 전체 예제를 참조하세요.

추가 정보

- Amazon CloudWatch 사용 설명서의 Amazon CloudWatch 경보 사용
- Amazon CloudWatch API 참조의 PutMetricAlarm
- Amazon CloudWatch API 참조의 DescribeAlarms
- Amazon CloudWatch API 참조의 DeleteAlarms

Amazon CloudWatch Events 사용

CloudWatch Events는 Amazon EC2 인스턴스, Lambda 함수, Kinesis 스트림, Amazon ECS 작업, Step Functions 상태 시스템, Amazon SNS 주제, Amazon SQS 대기열 또는 기본 제공 대상에 대한 AWS 리소스의 변경 사항을 설명하며 실시간에 가까운 시스템 이벤트 스트림을 제공합니다. 단순 규칙을 사용하여 일치하는 이벤트를 검색하고 하나 이상의 대상 함수 또는 스트림으로 이를 라우팅할 수 있습니다.

Amazon EventBridge는 CloudWatch Events의 업그레이드 버전입니다. 두 서비스 모두 동일한 API를 사용하므로 SDK에서 제공하는 CloudWatch Events 클라이언트를 계속 사용하거나 CloudWatch Events 기능을 위해 SDK for Java의 EventBridge 클라이언트로 마이그레이션할 수 있습니다. 이제 EventBridge 설명서 사이트를 통해 CloudWatch Events 사용 설명서 및 API 참조를 사용할 수 있습니다.

이벤트 추가

사용자 지정 CloudWatch 이벤트를 추가하려면 각 이벤트에 대한 세부 정보를 제공하는 하나 이상의 `PutEventsRequest` 객체가 포함된 `PutEventsRequestEntry` 객체를 사용하여 `CloudWatchEventsClient`'s `putEvents` 메서드를 호출합니다. 이벤트 유형 및 소스, 이벤트와 연결된 리소스 등 입력 항목에 대한 여러 파라미터를 지정할 수 있습니다.

> **Note**
>
> `putEvents` 호출당 최대 10개 이벤트를 지정할 수 있습니다.

가져옵니다.

```
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;
```

코드

```
    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);
        }
    }
```

GitHub의 전체 예제를 참조하세요.

규칙 추가

규칙을 만들거나 업데이트하려면 규칙 이름과 선택 사항 파라미터(예: 이벤트 패턴, 규칙과 연결되는 IAM 역할, 규칙 실행 빈도를 설명하는 일정 수식)를 포함한 `PutRuleRequest`를 사용하여 `CloudWatchEventsClient`'s `putRule` 메서드를 호출합니다.

가져옵니다.

```
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;
```

코드

```
    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);
        }
    }
```

GitHub의 전체 예제를 참조하세요.

대상 추가

대상은 규칙이 트리거될 때 호출되는 리소스입니다. 대상의 예로는 Amazon EC2 인스턴스, Lambda 함수, Kinesis 스트림, Amazon ECS 작업, Step Functions 상태 시스템 및 기본 제공 대상이 있습니다.

규칙에 대상을 추가하려면 업데이트할 규칙과 규칙에 추가할 대상 목록이 포함된 `PutTargetsRequest`를 사용하여 `CloudWatchEventsClient's putTargets` 메서드를 호출합니다.

가져옵니다.

```
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;
```

코드

```
    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);
        }
    }
```

GitHub의 전체 예제를 참조하세요.

추가 정보

- Amazon EventBridge 사용 설명서의 PutEvents와 함께 이벤트 추가
- Amazon EventBridge 사용 설명서의 규칙에 대한 스케줄 표현식
- Amazon EventBridge 사용 설명서의 CloudWatch Events 이벤트의 이벤트 유형
- Amazon EventBridge 사용 설명서의 이벤트 패턴
- Amazon EventBridge API 참조의 PutEvents
- Amazon EventBridge API 참조의 PutTargets
- Amazon EventBridge API 참조의 PutRule

AWS 데이터베이스 서비스 및 AWS SDK for Java 2.x

AWS는 관계형, 키-값, 인메모리, 문서 및 기타 여러 데이터베이스 유형을 제공합니다. Java 2.x용 SDK 지원은 AWS에서 데이터베이스 서비스의 특성에 따라 달라집니다.

Amazon DynamoDB 서비스와 같은 일부 데이터베이스 서비스에는 AWS 리소스(데이터베이스)를 관리하는 웹 서비스 API와 데이터와 상호 작용하는 웹 서비스 API가 있습니다. Java 2.x용 SDK에서 이러한 유형의 서비스에는 DynamoDBClient와 같은 전용 서비스 클라이언트가 있습니다.

다른 데이터베이스 서비스에는 Amazon DocumentDB API(클러스터, 인스턴스 및 리소스 관리용)와 같이 리소스와 상호 작용하는 웹 서비스 API가 있지만 데이터 작업을 위한 웹 서비스 API는 없습니다. Java 2.x용 SDK에는 리소스 작업을 위한 해당 DocDbClient 인터페이스가 있습니다. 그러나 데이터를 처리하려면 Java용 MongoDB와 같은 다른 Java API가 필요합니다.

아래 예를 사용하여 다양한 유형의 데이터베이스와 함께 Java 2.x 서비스 클라이언트용 SDK를 사용하는 방법을 알아보세요.

Amazon DynamoDB 예제

데이터 작업	데이터베이스 작업
SDK service client: DynamoDbClient Example: DynamoDB를 사용하는 React 및 Spring REST 애플리케이션 Examples: 여러 DynamoDB 예제	SDK service client: DynamoDbClient Examples: CreateTable, ListTables, DeleteTable
SDK service client: DynamoDbEnhancedClient Example: DynamoDB를 사용하는 React 및 Spring REST 애플리케이션 Examples: 여러 DynamoDB 예제 (names starting with 'Enhanced")	

이 가이드의 가이드 코드 예제 단원에서 추가 DynamoDB 예제를 참조하세요.

Amazon RDS 예제

데이터 작업	데이터베이스 작업
비 SDK API: JDBC, 데이터베이스별 SQL 버전. 사용자 코드가 데이터베이스 연결 또는 연결 풀을 관리합니다.	SDK 서비스 클라이언트: RdsClient
예: MySQL을 사용한 React 및 Spring REST 애플리케이션	예: 여러 RdsClient 예제

Amazon Redshift 예제

데이터 작업	데이터베이스 작업
SDK 서비스 클라이언트: RedshiftDataClient	SDK 서비스 클라이언트: RedshiftClient
예: 여러 RedshiftDataClient 예제	예: 여러 RedshiftClient 예제
예: RedshiftDataClient를 사용하는 React 및 Spring REST 애플리케이션	

Amazon Aurora Serverless v2 예제

데이터 작업	데이터베이스 작업
SDK 서비스 클라이언트: RdsDataClient	SDK 서비스 클라이언트: RdsClient
예: RdsDataClient를 사용하는 React 및 Spring REST 애플리케이션	예: 여러 RdsClient 예제

Amazon DocumentDB 예제

데이터 작업	데이터베이스 작업
비 SDK API: MongoDB 전용 자바 라이브러리(예: Java용 MongoDB). 사용자 코드가 데이터베이스 연결 또는 연결 풀을 관리합니다.	SDK 서비스 클라이언트: DocDbClient
예: DocumentDB(Mongo) 개발자 가이드 ('Java' 탭 선택)	

DynamoDB 작업

DynamoDB는 완전관리형 NoSQL 데이터베이스 서비스로, 원활한 확장성과 함께 빠르고 예측 가능한 성능을 제공합니다. 이 섹션에서는 AWS SDK for Java 2.x를 사용하여 DynamoDB로 작업하는 방법을 보여줍니다.

DynamoDB 클라이언트 선택

SDK는 DynamoDB를 사용한 작업을 위한 2가지 주요 접근 방식을 제공합니다.

하위 수준 클라이언트(`DynamoDbClient`)

요청 및 응답을 완전히 제어할 수 있는 DynamoDB 작업에 대한 직접 액세스를 제공합니다. 세분화된 제어가 필요하거나 동적 스키마로 작업하는 경우 이 클라이언트를 사용합니다.

향상된 클라이언트(`DynamoDbEnhancedClient`)

Java 객체와 DynamoDB 항목 간의 자동 매핑을 통해 객체 중심 프로그래밍을 제공합니다. 또한 고정된 스키마를 따르지 않는 JSON과 유사한 데이터 작업을 위한 문서 중심 기능을 제공합니다. 잘 정의된 데이터 모델 또는 문서 유형 데이터로 작업할 때 이 클라이언트를 사용합니다.

DynamoDB 클라이언트 구성

DynamoDB로 작업하기 전에 최적의 성능과 신뢰성을 위해 클라이언트를 구성합니다.

DynamoDB 재시도 동작 이해

DynamoDB 클라이언트는 다른 AWS 서비스 클라이언트보다 많은 기본 최대 재시도 횟수인 8을 사용합니다. 이렇게 재시도 횟수가 많으면 DynamoDB의 분산 특성 및 임시 용량 제한을 처리하는 데 도움이 됩니다. 재시도 전략에 대한 자세한 내용은 the section called “Retries” 섹션을 참조하세요.

계정 기반 엔드포인트로 성능 최적화

DynamoDB는 AWS 계정 ID를 사용하여 요청 라우팅을 간소화해 성능을 개선하는 AWS 계정 기반 엔드포인트를 제공합니다.

이 기능을 사용하려면 버전 2.28.4 이상의 AWS SDK for Java 2.x가 필요합니다. Maven 중앙 리포지토리에서 최신 버전을 확인해 보세요. 지원되는 SDK 버전은 자동으로 새 엔드포인트를 사용합니다.

계정 기반 라우팅을 옵트아웃하려면 다음 옵션 중 하나를 선택합니다.

- `AccountIdEndpointMode`를 `DISABLED`로 설정하여 DynamoDB 서비스 클라이언트를 구성합니다.
- 환경 변수를 설정합니다.
- JVM 시스템 속성을 설정합니다.
- 공유 AWS 구성 파일 설정을 업데이트합니다.

다음 예제에서는 DynamoDB 서비스 클라이언트를 구성하여 계정 기반 라우팅을 사용 해제하는 방법을 보여줍니다.

```
DynamoDbClient.builder()
                .accountIdEndpointMode(AccountIdEndpointMode.DISABLED)
                .build();
```

다른 구성 옵션에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서의 계정 기반 엔드포인트를 참조하세요.

이 주제에서 다루는 내용

다음 섹션에서는 DynamoDB로 작업하는 방법을 보여줍니다.

- the section called “DynamoDB의 테이블 다루기” - 테이블을 만들고, 설명, 업데이트 및 삭제
- the section called “의 항목 작업 DynamoDB” - 개별 항목 추가, 검색 및 업데이트

- the section called “ 객체를 DynamoDB 항목에 매핑” - 향상된 클라이언트에서 객체 매핑 및 문서 중심 데이터 사용

추가 DynamoDB 코드 예제는 AWS 코드 예제 라이브러리의 DynamoDB 코드 예제를 참조하세요.

DynamoDB의 테이블 다루기

테이블은 DynamoDB 데이터베이스에 있는 모든 항목의 컨테이너입니다. DynamoDB에 데이터를 추가하거나 삭제하기 전에 먼저 테이블을 만들어야 합니다.

각 테이블마다 다음을 정의해야 합니다.

- 계정 및 리전에 고유한 테이블 이름입니다.
- 모든 값이 고유한 기본 키. 테이블의 두 항목에 동일한 기본 키 값을 지정할 수 없습니다.

  기본 키는 단일 파티션(HASH) 키로 이루어진 단순형이거나, 파티션과 정렬(RANGE) 키로 이루어진 복합형일 수 있습니다.

  각 키 값에는 `ScalarAttributeType` 클래스에 따라 열거되는 관련 데이터 유형이 있습니다. 키 값은 이진(B), 숫자(N) 또는 문자열(S)일 수 있습니다. 자세한 정보는 Amazon DynamoDB 개발자 안내서의 명명 규칙과 데이터 유형을 참조하세요.
- 프로비저닝된 처리량은 테이블에 예약된 읽기/쓰기 용량 단위 수를 정의하는 값입니다.

> **Note**
>
> Amazon DynamoDB 요금은 테이블에 대해 설정하는 프로비저닝된 처리량 값을 기준으로 하므로 테이블에 필요하다고 생각되는 만큼의 용량만 예약하세요.

  언제라도 테이블의 프로비저닝된 처리량을 수정할 수 있으므로 변경이 필요할 경우 용량을 조정할 수 있습니다.

테이블 생성

`DynamoDbClient`’s `createTable` 메서드를 사용하여 새 DynamoDB 테이블을 만듭니다. 테이블 속성과 테이블 스키마를 구성해야 하며, 이 두 가지 요소 모두 테이블의 기본 키를 식별하는 데 사용됩니다. 또한 프로비저닝된 초기 처리량 값과 테이블 이름도 지정해야 합니다.

> **Note**
>
> 선택한 이름의 테이블이 이미 있는 경우 DynamoDbException이 발생합니다.

단순형 기본 키를 사용하여 테이블 생성

이 코드는 테이블의 단순 프라이머리 키라는 속성 하나를 포함하는 테이블을 만듭니다. 이 예제에서는 CreateTableRequest에 AttributeDefinition 및 KeySchemaElement 객체를 사용합니다.

가져옵니다.

```
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;
```

코드

```
    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 "";
    }
```

GitHub의 전체 예제를 참조하세요.

복합형 기본 키를 사용하여 테이블 생성

다음 예제는 두 가지 속성이 있는 테이블을 만듭니다. 두 속성 모두 복합 기본 키에 사용됩니다.

가져옵니다.

```
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;
```

코드

```
    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 "";
    }
```

GitHub의 전체 예제를 참조하세요.

테이블 나열

DynamoDbClient's listTables 메서드를 호출하여 특정 리전의 테이블을 나열할 수 있습니다.

> **Note**
>
> 계정 및 리전에 대해 이름이 지정된 테이블이 없으면 ResourceNotFoundException이 발생합니다.

가져옵니다.

```
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;
```

코드

```
    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!");
    }
```

기본적으로 호출당 최대 100개의 테이블이 반환됩니다. 반환된 `ListTablesResponse` 객체에 `lastEvaluatedTableName`을 사용하여 마지막으로 평가된 테이블을 가져올 수 있습니다. 이 값을 사용하여 이전 목록의 마지막으로 반환된 값 다음에 이어지는 목록을 시작할 수 있습니다.

GitHub의 전체 예제를 참조하세요.

테이블 설명(테이블에 대한 정보 가져오기)

테이블에 대한 정보를 가져오려면 `DynamoDbClient`'s `describeTable` 메서드를 사용합니다.

> **Note**
>
> 계정 및 리전에 대해 이름이 지정된 테이블이 없으면 `ResourceNotFoundException`이 발생합니다.

가져옵니다.

```
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;
```

코드

```
    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!");
    }
```

GitHub의 전체 예제를 참조하세요.

테이블 수정(업데이트)

언제라도 DynamoDbClient's updateTable 메서드를 호출하여 테이블의 프로비저닝된 처리량 값을 수정할 수 있습니다.

> **Note**
>
> 계정 및 리전에 대해 이름이 지정된 테이블이 없으면 ResourceNotFoundException이 발생합니다.

가져옵니다.

```
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;
```

코드

```
    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!");
    }
```

GitHub의 전체 예제를 참조하세요.

테이블 삭제

테이블을 삭제하려면 `DynamoDbClient`'s `deleteTable` 메서드를 호출하고 테이블 이름을 제공하세요.

> **Note**
>
> 계정 및 리전에 대해 이름이 지정된 테이블이 없으면 ResourceNotFoundException이 발생합니다.

가져옵니다.

```
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;
```

코드

```
    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!");
    }
```

GitHub의 전체 예제를 참조하세요.

추가 정보

- Amazon DynamoDB 개발자 안내서의 테이블 작업 지침
- Amazon DynamoDB 개발자 안내서의 DynamoDB의 테이블 다루기

의 항목 작업 DynamoDB

에서 DynamoDB항목은 속성 모음으로, 각 속성에는 이름과 값이 있습니다. 속성 값은 스칼라, 세트 또는 문서 유형일 수 있습니다. 자세한 정보는 Amazon DynamoDB 개발자 안내서의 명명 규칙과 데이터 유형을 참조하세요.

테이블에서 항목 검색(가져오기)

DynamoDbClient의 `getItem` 메서드를 호출하여 테이블 이름과 원하는 항목의 기본 키 값이 포함된 GetItemRequest 객체를 이 메서드에 전달합니다. GetItemResponse 객체와 해당되는 모든 속성을 반환합니다. 특정 속성을 검색하기 위해 `GetItemRequest`에서 하나 이상의 프로젝션 표현식을 지정할 수 있습니다.

반환된 GetItemResponse 객체의 item( ) 메서드를 사용하여 항목과 연결된 키(문자열) 및 값 (AttributeValue) 쌍의 맵을 가져올 수 있습니다.

가져오기

```
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;
```

코드

```
   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);
        }
    }
```

GitHub의 전체 예제를 참조하세요.

비동기 클라이언트를 사용하여 테이블에서 항목 검색(가져오기)

DynamoDbAsyncClient의 `getItem` 메서드를 호출하여 테이블 이름과 원하는 항목의 기본 키 값이 포함된 GetItemRequest 객체를 이 메서드에 전달합니다.

해당 항목에 대한 모든 속성이 포함된 Collection 인스턴스를 반환할 수 있습니다(다음 예제 참조).

가져오기

```
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;
```

코드

```
   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);
        }
```

GitHub의 전체 예제를 참조하세요.

테이블에 새 항목 추가

항목의 속성을 나타내는 키-값 페어의 맵을 생성합니다. 여기에는 테이블의 기본 키 필드 값이 포함되어야 합니다. 기본 키로 식별되는 항목이 이미 존재하면 필드가 요청에 따라 업데이트됩니다.

> **Note**
>
> 계정 및 리전에 대해 이름이 지정된 테이블이 없으면 ResourceNotFoundException이 발생합니다.

가져오기

```
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;
```

코드

```
    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);
```

```
        }
    }
```

GitHub의 전체 예제를 참조하세요.

테이블의 기존 항목 업데이트

DynamoDbClient의 `updateItem` 메서드를 사용하여 테이블 이름, 기본 키 값 및 업데이트할 필드 맵을 지정함으로써 테이블에 이미 존재하는 항목의 속성을 업데이트할 수 있습니다.

> **Note**
>
> 해당 계정 및 리전에 대해 이름이 지정된 테이블이 없거나 전달한 기본 키로 식별되는 항목이 없으면 ResourceNotFoundException이 발생합니다.

가져오기

```
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;
```

코드

```
    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!");
    }
```

GitHub의 전체 예제를 참조하세요.

테이블의 기존 항목 삭제

DynamoDbClient의 `deleteItem` 메서드를 사용하고 기본 키 값 및 테이블 이름을 지정하여 테이블에 있는 항목을 삭제할 수 있습니다.

> **Note**
>
> 해당 계정 및 리전에 대해 이름이 지정된 테이블이 없거나 전달한 기본 키로 식별되는 항목이 없으면 ResourceNotFoundException이 발생합니다.

가져오기

```
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;
```

코드

```
 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);
       }
   }
```

GitHub의 전체 예제를 참조하세요.

추가 정보

- Amazon DynamoDB 개발자 안내서의 항목 작업 지침
- Amazon DynamoDB 개발자 안내서의 에서 항목 작업 DynamoDB

AWS SDK for Java 2.x를 사용하여 Java 객체를 DynamoDB 항목에 매핑

DynamoDB 향상된 클라이언트 API는 Java v1.x용 SDK `DynamoDBMapper` 클래스의 후속 클래스인 상위 수준 라이브러리입니다. 클라이언트 측 클래스를 DynamoDB 테이블에 매핑하는 간단한 방법을 제공합니다. 코드에서 테이블과 해당 데이터 클래스 간의 관계를 정의합니다. 이러한 관계를 정의한 다음 DynamoDB의 테이블 또는 항목에 대해 다양한 생성, 읽기, 업데이트 또는 삭제(CRUD) 작업을 직관적으로 수행할 수 있습니다.

DynamoDB 향상된 클라이언트 API에는 정의된 스키마를 따르지 않는 문서 유형 항목을 처리할 수 있는 향상된 클라이언트 API도 포함되어 있습니다.

DynamoDB 향상된 클라이언트 API는 다음 항목에서 설명합니다.

- DynamoDB 향상된 클라이언트 API를 사용하여 시작하기
- DynamoDB 향상된 클라이언트 API의 기본 사항 알아보기
- 고급 매핑 기능 사용
- DynamoDB용 향상된 문서 API를 사용하여 JSON 문서로 작업
- 확장 프로그램을 사용하여 DynamoDB 향상된 클라이언트 작업 사용자 지정
- 비동기식으로 DynamoDB 향상된 클라이언트 API 사용
- 데이터 클래스 주석

DynamoDB 향상된 클라이언트 API를 사용하여 시작하기

다음 자습서에서는 DynamoDB 향상된 클라이언트 API를 사용하는 데 필요한 기본 사항을 소개합니다.

종속성 추가

프로젝트에서 DynamoDB 향상된 클라이언트 API로 작업을 시작하려면 `dynamodb-enhanced` Maven 아티팩트에 종속성을 추가하세요. 방법은 다음 예제와 같습니다.

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>
```

Maven 중앙 리포지토리에서 최신 버전을 검색하고 *<VERSION>*을 이 값으로 바꾸세요.

Gradle

```
repositories {
    mavenCentral()
}
dependencies {
    implementation(platform("software.amazon.awssdk:bom:<VERSION>"))
    implementation("software.amazon.awssdk:dynamodb-enhanced")
    ...
}
```

Maven 중앙 리포지토리에서 최신 버전을 검색하고 *<VERSION>*을 이 값으로 바꾸세요.

데이터 클래스에서 **TableSchema** 생성

`TableSchema`를 사용하면 향상된 클라이언트가 DynamoDB 속성 값을 클라이언트 측 클래스와 매핑하거나 클라이언트 측 클래스에서 가져올 수 있습니다. 이 자습서에서는 정적 데이터 클래스에서 파생되고 빌더를 사용하여 코드에서 생성되는 `TableSchema`에 대해 알아봅니다.

주석이 달린 데이터 클래스 사용하기

Java 2.x용 SDK에는 데이터 클래스와 함께 신속하게 `TableSchema`를 생성하는 데 사용할 수 있는 일련의 주석이 포함되어 있습니다.

먼저 JavaBean 사양을 준수하는 데이터 클래스를 만드세요. 이 사양에서는 클래스에 인수가 없는 공용 생성자가 있어야 하고 클래스의 각 속성에 대한 접근자와 설정자가 있어야 합니다. 데이터 클래스가 DynamoDbBean임을 나타내는 클래스 수준 주석을 포함하세요. 또한 최소한 접근자 또는 설정자에 기본 키 속성에 대한 DynamoDbPartitionKey 주석을 포함해야 합니다.

속성 수준 주석을 getter 또는 setter에 적용할 수 있지만 둘 다 적용할 수는 없습니다.

> **Note**
>
> 용어 property는 일반적으로 JavaBean에서 캡슐화된 값에 사용됩니다. 하지만 이 가이드에서는 DynamoDB에서 사용하는 용어와의 일관성을 위해 용어 attribute를 대신 사용합니다.

다음 Customer 클래스는 클래스 정의를 DynamoDB 테이블에 연결하는 주석을 보여줍니다.

Customer 클래스

```
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 + "]";
    }
}
```

주석이 달린 데이터 클래스를 만든 후 다음 코드 조각에 표시된 것처럼 이 클래스를 사용하여 `TableSchema`를 생성합니다.

```
static final TableSchema<Customer> customerTableSchema =
 TableSchema.fromBean(Customer.class);
```

`TableSchema`는 정적이고 변경할 수 없도록 설계되었습니다. 일반적으로 클래스 로드 시 인스턴스화할 수 있습니다.

정적 `TableSchema.fromBean()` 팩토리 메서드는 bean을 검사하여 DynamoDB 속성과 데이터 클래스 속성 간의 매핑을 생성합니다.

여러 데이터 클래스로 구성된 데이터 모델을 사용하는 예제는 ??? 단원의 `Person` 클래스를 참조하세요.

빌더 사용

코드에 테이블 스키마를 정의하면 Bean 인트로스펙션 비용을 건너뛸 수 있습니다. 스키마를 코딩하면 클래스가 JavaBean 이름 지정 표준을 따르지 않아도 되며 주석을 달 필요도 없습니다. 다음 예제는 빌더를 사용하며 주석을 사용하는 `Customer` 클래스 예제와 동일합니다.

```
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();
```

향상된 클라이언트를 만들고 **`DynamoDbTable`**

확장 클라이언트 생성

DynamoDB 향상된 클라이언트 클래스 또는 이에 상응하는 비동기식 DynamoDbEnhancedAsyncClient는 DynamoDB 향상된 클라이언트 API 사용을 위한 시작점입니다.

향상된 클라이언트에는 작업을 수행하기 위한 표준 `DynamoDbClient`이 필요합니다. API는 `DynamoDbEnhancedClient` 인스턴스를 생성하는 두 가지 방법을 제공합니다. 다음 코드 조각에 표시된 첫 번째 옵션은 구성 설정에서 기본 설정을 선택하여 표준 `DynamoDbClient`을 생성합니다.

```
DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.create();
```

기본 표준 클라이언트를 구성하려는 경우 다음 코드 조각과 같이 확장 클라이언트의 빌더 메서드에 제공할 수 있습니다.

```
// 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();
```

DynamoDbTable 인스턴스 생성

DynamoDbTable를 TableSchema에서 제공하는 매핑 기능을 사용하는 DynamoDB 테이블의 클라이언트 측 표현이라고 생각하면 됩니다. DynamoDbTable 클래스는 단일 DynamoDB 테이블과 상호 작용할 수 있는 CRUD 작업을 위한 메서드를 제공합니다.

DynamoDbTable<T>는 문서 유형 항목으로 작업할 때 사용자 정의 클래스이든 EnhancedDocument이든 관계없이 단일 형식 인수를 사용하는 제네릭 클래스입니다. 이 인수 유형은 사용하는 클래스와 단일 DynamoDB 테이블 간의 관계를 설정합니다.

DynamoDbEnhancedClient의 table() 팩토리 메서드를 사용하여 다음 코드 조각과 같이 DynamoDbTable 인스턴스를 생성합니다.

```
static final DynamoDbTable<Customer> customerTable =
        enhancedClient.table("Customer", TableSchema.fromBean(Customer.class));
```

DynamoDbTable 인스턴스는 변경이 불가능하고 애플리케이션 전체에서 사용할 수 있기 때문에 싱글톤에 적합합니다.

이제 코드에 Customer 인스턴스로 작업할 수 있는 DynamoDB 테이블의 인메모리 표현이 생겼습니다. 실제 DynamoDB 테이블은 존재할 수도 있고 존재하지 않을 수도 있습니다. 이름이 Customer로 지정된 테이블이 이미 있는 경우 해당 테이블에 대해 CRUD 작업을 시작할 수 있습니다. 테이블이 존재하지 않는 경우 다음 단원에서 설명하는 대로 DynamoDbTable 인스턴스를 사용하여 테이블을 생성하세요.

필요한 경우 DynamoDB 테이블 생성

DynamoDbTable 인스턴스를 생성한 후 이를 사용하여 DynamoDB에서 테이블을 일회성으로 생성합니다.

테이블 생성 예제 코드

다음 예제는 Customer 데이터 클래스를 기반으로 DynamoDB 테이블을 생성합니다.

이 예제는 클래스 이름과 동일한 이름 Customer을 가진 DynamoDB 테이블을 생성하지만 테이블 이름은 다른 이름일 수 있습니다. 테이블 이름을 무엇으로 지정하든 테이블 작업을 하려면 추가 애플리케이션에서 이 이름을 사용해야 합니다. 기본 DynamoDB 테이블을 사용하기 위해 다른 DynamoDbTable 객체를 생성할 때마다 이 이름을 table() 메서드에 제공하세요.

`createTable` 메서드에 전달된 Java Lambda 파라미터 `builder`를 사용하면 테이블을 사용자 지정할 수 있습니다. 이 예시에서는 프로비저닝된 처리량을 구성합니다. 테이블을 생성할 때 기본 설정을 사용하려면 다음 코드 조각과 같이 빌더를 건너뛰세요.

```
customerTable.createTable();
```

기본 설정을 사용하는 경우 프로비저닝된 처리량 값은 설정되지 않습니다. 대신 테이블의 청구 모드가 온디맨드로 설정됩니다.

또한 이 예제에서는 응답에서 받은 테이블 이름을 출력하려고 시도하기 전에 `DynamoDbWaiter`를 사용합니다. 테이블을 만드는 데 시간이 좀 걸립니다. 따라서 웨이터를 사용하면 테이블을 사용하기 전에 DynamoDB 서비스를 폴링하여 테이블이 존재하는지 확인하는 로직을 작성할 필요가 없습니다.

가져오기

```
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.enhanced.dynamodb.model.CreateTableEnhancedRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;
```

코드

```
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**
>
> 데이터 클래스에서 테이블을 생성할 때 DynamoDB 테이블의 속성 이름은 소문자로 시작합니다. 테이블의 속성 이름을 대문자로 시작하려면 `@DynamoDbAttribute(NAME)` 주석을 사용하고 원하는 이름을 파라미터로 제공하세요.

작업을 수행

테이블이 생성된 후 `DynamoDbTable` 인스턴스를 사용하여 DynamoDB 테이블에 대한 작업을 수행합니다.

다음 예제에서는 `DynamoDbTable<Customer>` 싱글톤이 `Customer`데이터 클래스 인스턴스와 함께 파라미터로 전달되어 테이블에 새 항목을 추가합니다.

```
   public static void putItemExample(DynamoDbTable<Customer> customerTable, Customer
customer){
       logger.info(customer.toString());
       customerTable.putItem(customer);
   }
```

`Customer` 객체

```
       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"));
```

customer 객체를 DynamoDB 서비스에 보내기 전에 객체 toString() 메서드의 출력을 기록하여 향상된 클라이언트가 보내는 것과 비교하세요.

```
Customer [id=1, name=Customer Name, email=customer@example.com,
 regDate=2023-07-03T10:15:30Z]
```

와이어 레벨 로깅은 생성된 요청의 페이로드를 보여줍니다. 향상된 클라이언트는 데이터 클래스에서 저수준 표현을 생성했습니다. Java의 Instant 유형인 regDate 속성은 DynamoDB 문자열로 표시됩니다.

```
{
  "TableName": "Customer",
  "Item": {
    "registrationDate": {
      "S": "2023-07-03T10:15:30Z"
    },
    "id": {
      "S": "1"
    },
    "custName": {
      "S": "Customer Name"
    },
    "email": {
      "S": "customer@example.com"
    }
  }
}
```

기존 테이블로 작업하기

이전 단원에서는 Java 데이터 클래스로 시작하는 DynamoDB 테이블을 생성하는 방법을 살펴보았습니다. 기존 테이블이 이미 있고 향상된 클라이언트의 기능을 사용하려는 경우 해당 테이블과 함께 사용할 Java 데이터 클래스를 생성할 수 있습니다. DynamoDB 테이블을 검사하고 데이터 클래스에 필요한 주석을 추가해야 합니다.

기존 테이블로 작업하기 전에 DynamoDbEnhanced.table() 메서드를 호출하세요. 이 작업은 이전 예제에서 다음 명령문을 사용하여 수행되었습니다.

```
DynamoDbTable<Customer> customerTable = enhancedClient.table("Customer",
 TableSchema.fromBean(Customer.class));
```

DynamoDbTable 인스턴스가 반환되면 기본 테이블을 사용하여 바로 작업을 시작할 수 있습니다. DynamoDbTable.createTable() 메서드를 호출하여 테이블을 다시 만들 필요는 없습니다.

다음 예제는 DynamoDB 테이블에서 Customer 인스턴스를 즉시 가져와서 이를 보여줍니다.

```
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**
>
> table() 메서드에 사용된 테이블 이름은 기존 DynamoDB 테이블 이름과 일치해야 합니다.

DynamoDB 향상된 클라이언트 API의 기본 사항 알아보기

이 항목에서는 DynamoDB 향상된 클라이언트 API의 기본 기능을 설명하고 이를 표준 DynamoDB 클라이언트 API와 비교합니다.

DynamoDB 향상된 클라이언트 API를 처음 사용하는 경우 입문 자습서를 통해 기본 클래스를 익히는 것이 좋습니다.

Java에서 DynamoDB 항목

DynamoDB 테이블은 항목을 저장합니다. 사용 사례에 따라 Java 측 항목은 정적으로 구조화된 데이터 또는 동적으로 생성된 구조의 형태를 취할 수 있습니다.

사용 사례에서 일관된 속성 집합을 가진 항목을 요구하는 경우 주석이 달린 클래스를 사용하거나 빌더를 사용하여 적절한 정적 형식의 TableSchema을 생성하세요.

또는 다양한 구조로 구성된 항목을 저장해야 하는 경우 DocumentTableSchema를 생성하세요. DocumentTableSchema는 향상된 문서 API 일부이며 정적으로 입력된 기본 키만 필요하며

EnhancedDocument 인스턴스와 함께 작동하여 데이터 요소를 보관합니다. 향상된 문서 API에 대해서는 다른 항목에서 다룹니다.

데이터 모델 클래스의 속성 유형

DynamoDB는 Java의 다양한 형식 시스템에 비해 적은 수의 속성 유형을 지원하지만 DynamoDB 향상된 클라이언트 API는 Java 클래스의 멤버를 DynamoDB 속성 유형으로 또는 DynamoDB 속성 유형에서 변환하는 메커니즘을 제공합니다.

Java 데이터 클래스의 속성 유형은 프리미티브가 아닌 객체 유형이어야 합니다. 예를 들어 항상 long 및 int 프리미티브가 아닌 Long 및 Integer 객체 데이터 형식을 사용합니다.

기본적으로 DynamoDB 향상된 클라이언트 API는 정수, 문자열, BigDecimal, and 인스턴트같은 다양한 유형에 대한 속성 변환기를 지원합니다. 목록은 AttributeConverter 인터페이스의 알려진 구현 클래스에 표시됩니다. 목록에는 지도, 목록, 세트 등 다양한 유형과 컬렉션이 포함됩니다.

기본적으로 지원되지 않거나 JavaBean 규칙을 준수하지 않는 속성 유형에 대한 데이터를 저장하려면 사용자 정의 AttributeConverter 구현을 작성하여 변환을 수행할 수 있습니다. 예제는 속성 변환 단원을 참조하세요.

클래스가 Java Beans 사양을 준수하는 속성 유형(또는 변경할 수 없는 데이터 클래스)의 데이터를 저장하려면 두 가지 방법을 사용할 수 있습니다.

- 소스 파일에 액세스할 수 있는 경우 @DynamoDbBean(또는 @DynamoDbImmutable)로 클래스에 주석을 달 수 있습니다. 중첩 속성에 대해 설명하는 단원에서는 주석이 달린 클래스를 사용하는 예제를 보여줍니다.

- 속성에 대한 JavaBean 데이터 클래스의 소스 파일에 대한 액세스 권한이 없는 경우(또는 액세스 권한이 있는 클래스의 소스 파일에 주석을 달고 싶지 않은 경우) 빌더 접근 방식을 사용할 수 있습니다. 이렇게 하면 키를 정의하지 않고 테이블 스키마가 생성됩니다. 그런 다음 이 테이블 스키마를 다른 테이블 스키마에 중첩하여 매핑을 수행할 수 있습니다. 중첩 속성 단원에는 중첩 스키마 사용을 보여주는 예제가 있습니다.

Null 값

putItem 메서드를 사용할 때 향상된 클라이언트는 매핑된 데이터 객체의 null 값 속성을 DynamoDB에 대한 요청에 포함하지 않습니다.

updateItem 요청에 대한 SDK의 기본 동작은 updateItem 메서드에서 제출하는 객체에서 null로 설정된 DynamoDB의 항목에서 속성을 제거합니다. 일부 속성 값을 업데이트하고 다른 속성 값을 변경하지 않으려면 2가지 옵션을 참고할 수 있습니다.

- 값을 변경하기 전에 항목을 검색합니다(getItem 사용). SDK는 이 접근 방식을 사용하여 모든 업데이트된 값과 이전 값을 DynamoDB에 제출합니다.
- 항목을 업데이트하기 위한 요청을 구축할 때 IgnoreNullsMode.SCALAR_ONLY 또는 IgnoreNullsMode.MAPS_ONLY를 사용합니다. 두 모드 모두 DynamoDB의 스칼라 속성을 나타내는 객체의 null 값 속성을 무시합니다. 이 가이드의 the section called “복잡한 유형이 포함된 항목 업데이트” 주제에는 IgnoreNullsMode 값에 대한 자세한 내용과 복잡한 유형으로 작업하는 방법이 포함되어 있습니다.

다음 예제는 updateItem() 메서드의 ignoreNullsMode()를 보여줍니다.

```
    public static void updateItemNullsExample() {
        Customer customer = new Customer();
        customer.setCustName("CustomerName");
        customer.setEmail("email");
        customer.setId("1");
        customer.setRegistrationDate(Instant.now());

        logger.info("Original customer: {}", customer);

        // Put item with values for all attributes.
        try {
            customerAsyncDynamoDbTable.putItem(customer).join();
        } catch (RuntimeException rte) {
            logger.error("A exception occurred during putItem: {}",
rte.getCause().getMessage(), rte);
            return;
        }

        // Create a Customer instance with the same 'id' and 'email' values, but a
different 'name' value.
        // Do not set the 'registrationDate' attribute.
        Customer customerForUpdate = new Customer();
        customerForUpdate.setCustName("NewName");
        customerForUpdate.setEmail("email");
        customerForUpdate.setId("1");
```

```
        // Update item without setting the 'registrationDate' property and set
 IgnoreNullsMode to SCALAR_ONLY.
        try {
            Customer updatedWithNullsIgnored = customerAsyncDynamoDbTable.updateItem(b
 -> b
                            .item(customerForUpdate)
                            .ignoreNullsMode(IgnoreNullsMode.SCALAR_ONLY))
                    .join();
            logger.info("Customer updated with nulls ignored: {}",
 updatedWithNullsIgnored.toString());
        } catch (RuntimeException rte) {
            logger.error("An exception occurred during updateItem: {}",
 rte.getCause().getMessage(), rte);
            return;
        }

        // Update item without setting the registrationDate attribute and not setting
 ignoreNulls to true.
        try {
            Customer updatedWithNullsUsed =
 customerAsyncDynamoDbTable.updateItem(customerForUpdate)
                    .join();
            logger.info("Customer updated with nulls used: {}",
 updatedWithNullsUsed.toString());
        } catch (RuntimeException rte) {
            logger.error("An exception occurred during updateItem: {}",
 rte.getCause().getMessage(), rte);
        }
    }


// Logged lines.
Original customer: Customer [id=1, name=CustomerName, email=email,
 regDate=2024-10-11T14:12:30.222858Z]
Customer updated with nulls ignored: Customer [id=1, name=NewName, email=email,
 regDate=2024-10-11T14:12:30.222858Z]
Customer updated with nulls used: Customer [id=1, name=NewName, email=email,
 regDate=null]
```

DynamoDB 향상된 클라이언트의 기본 메서드

향상된 클라이언트의 기본 메서드는 이름이 붙은 DynamoDB 서비스 작업에 매핑합니다. 다음 예제는 각 방법의 가장 간단한 변형을 보여줍니다. 향상된 요청 개체를 전달하여 각 메서드를 사용자 지정할

수 있습니다. 향상된 요청 객체는 표준 DynamoDB 클라이언트에서 사용할 수 있는 대부분의 기능을 제공합니다. 이는 AWS SDK for Java 2.x API 참조에 완전히 문서화되어 있습니다.

이 예제에서는 이전에 표시된 the section called “Customer 클래스”을 사용합니다.

```
// 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));
```

DynamoDB 향상된 클라이언트와 표준 DynamoDB 클라이언트 비교

표준 및 고급 DynamoDB 클라이언트 API를 모두 사용하면 DynamoDB 테이블을 사용하여 CRUD(생성, 읽기, 업데이트 및 삭제) 데이터 수준 작업을 수행할 수 있습니다. 클라이언트 API 간의 차이는 이를 수행하는 방법에 있습니다. 표준 클라이언트를 사용하면 저수준 데이터 속성으로 직접 작업할 수 있습니다. 향상된 클라이언트 API는 친숙한 Java 클래스를 사용하고 이면의 하위 수준 API에 매핑됩니다.

두 클라이언트 API 모두 데이터 수준 작업을 지원하지만 표준 DynamoDB 클라이언트는 리소스 수준 작업도 지원합니다. 리소스 수준 작업은 백업 생성, 테이블 나열, 테이블 업데이트와 같은 데이터베이스를 관리합니다. 향상된 클라이언트 API는 테이블 생성, 설명, 삭제와 같은 엄선된 리소스 수준 작업을 지원합니다.

두 클라이언트 API에서 사용하는 다양한 접근 방식을 설명하기 위해 다음 코드 예제는 표준 클라이언트와 향상된 클라이언트를 사용하여 동일한 `ProductCatalog` 테이블을 생성하는 방법을 보여줍니다.

비교: 표준 DynamoDB 클라이언트를 사용하여 테이블 생성

```
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))
);
```

비교: DynamoDB 향상된 클라이언트를 사용하여 테이블 생성

```
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))
        )
);
```

향상된 클라이언트는 다음과 같은 주석이 달린 데이터 클래스를 사용합니다. DynamoDB 향상된 클라이언트는 Java 데이터 형식을 DynamoDB 데이터 형식에 매핑하므로 코드가 복잡하지 않고 따라하기 쉽습니다. `ProductCatalog`는 DynamoDB 향상된 클라이언트에서 변경할 수 없는 클래스를 사용하는 예입니다. 매핑된 데이터 클래스에 변경할 수 없는 클래스를 사용하는 방법은 이 항목의 뒷부분에서 설명합니다.

`ProductCatalog` 클래스

```
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);
        }
    }
}
```

다음 두 개의 일괄 쓰기 코드 예제는 향상된 클라이언트와 달리 표준 클라이언트를 사용할 때 장황하고 형식 안전성이 부족하다는 것을 보여줍니다.

비교: 표준 DynamoDB 클라이언트를 사용하는 일괄 쓰기

```
   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());
    }
```

비교: DynamoDB 향상된 클라이언트를 사용하는 일괄 쓰기

```
    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());
    }
```

변경할 수 없는 데이터 클래스로 작업

DynamoDB 향상된 클라이언트 API의 매핑 기능은 변경할 수 없는 데이터 클래스와 함께 작동합니다. 불변 클래스에는 접근자만 포함되며 SDK가 클래스의 인스턴스를 생성하는 데 사용하는 빌더 클래스가 필요합니다. 변경 불가능한 클래스는 Customer 클래스에 표시된 `@DynamoDbBean` 주석을 사용하는 대신 사용할 빌더 클래스를 나타내는 매개변수를 사용하는 `@DynamoDbImmutable` 주석을 사용합니다.

다음 클래스는 `Customer`의 변경할 수 없는 버전입니다.

```
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); }
    }
}
```

`@DynamoDbImmutable`를 사용해 데이터 클래스에 주석을 달 때는 다음 요구 사항을 충족해야 합니다.

1. `Object.class`의 오버라이드도 아니고 `@DynamoDbIgnore`를 사용해 주석이 추가되지도 않은 모든 메서드는 DynamoDB 테이블의 속성에 대한 접근자여야 합니다.

2. 모든 접근자는 빌더 클래스에 해당하는 대소문자를 구분하는 설정자를 가져야 합니다.
3. 다음 시공 조건 중 하나만 충족해야 합니다.
   - 빌더 클래스에는 공개 기본 생성자가 있어야 합니다.
   - 데이터 클래스에는 매개 변수를 사용하지 않고 빌더 클래스의 인스턴스를 반환하는 이름이 `builder()`로 지정된 공용 정적 메서드가 있어야 합니다. 이 옵션은 변경할 수 없는 `Customer` 클래스에 표시됩니다.
4. 빌더 클래스에는 매개 변수를 사용하지 않고 변경 불가능한 클래스의 인스턴스를 반환하는 `build()`로 이름이 지정된 공용 메서드가 있어야 합니다.

변경할 수 없는 클래스를 위한 `TableSchema`를 만들려면 다음 코드 조각과 같이 `TableSchema`의 `fromImmutableClass()` 메서드를 사용하세요.

```
static final TableSchema<CustomerImmutable> customerImmutableTableSchema =
                          TableSchema.fromImmutableClass(CustomerImmutable.class);
```

변경 가능한 클래스에서 DynamoDB 테이블을 생성할 수 있는 것처럼, 다음 코드 조각 예제와 같이 `DynamoDbTable`의 `createTable()`를 한 번 호출하여 변경할 수 없는 클래스에서 테이블을 생성할 수 있습니다.

```
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));
}
```

Lombok과 같은 타사 라이브러리를 사용

Project Lombok과 같은 타사 라이브러리는 변경할 수 없는 객체와 관련된 보일러플레이트 코드를 생성하는 데 도움이 됩니다. DynamoDB 향상된 클라이언트 API는 데이터 클래스가 이 단원에 자세히 설명된 규칙을 따르는 한 이러한 라이브러리와 함께 작동합니다.

다음 예제에서는 Lombok 주석이 있는 변경 불가능한 `CustomerImmutable` 클래스를 보여줍니다. Lombok의 `onMethod` 기능이 속성 기반 DynamoDB 주석(예:`@DynamoDbPartitionKey`)을 생성된 코드에 복사하는 방법에 주목하세요.

```
@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;
}
```

표현식 및 조건 사용

DynamoDB 향상된 클라이언트 API의 표현식은 DynamoDB 식을 Java로 표현한 것입니다.

DynamoDB 향상된 클라이언트 API는 세 가지 유형의 식을 사용합니다.

`표현식`

`Expression` 클래스는 조건과 필터를 정의할 때 사용됩니다.

`QueryConditional`

이 유형의 표현식은 쿼리 작업의 주요 조건을 나타냅니다.

`UpdateExpression`

이 클래스는 DynamoDB 업데이트 표현식을 작성하는 데 도움이 되며, 현재 항목을 업데이트할 때 확장 프레임워크에서 사용됩니다.

표현 해부학

표현식은 다음과 같이 구성됩니다.

- 문자열 표현식(필수). 문자열에는 속성 이름 및 속성 값에 대한 자리 표시자 이름이 있는 DynamoDB 논리 표현식이 포함되어 있습니다.
- 표현식 값 맵(일반적으로 필수).
- 표현식 이름 맵(선택 사항).

빌더를 사용하여 다음과 같은 일반적인 형식을 취하는 `Expression` 객체를 생성합니다.

```
Expression expression = Expression.builder()
                            .expression(<String>)
                            .expressionNames(<Map>)
                            .expressionValues(<Map>)
                           .build()
```

`Expression`는 일반적으로 표현식 값의 맵이 필요합니다. 맵은 문자열 표현식의 자리 표시자 값을 제공합니다. 맵 키는 콜론(`:`) 이 붙은 자리 표시자 이름으로 구성되며 맵 값은 AttributeValue의 인스턴스입니다. AttributeValues 클래스에는 리터럴에서 `AttributeValue` 인스턴스를 생성할 수 있는 편리한 메서드가 있습니다. 또는 `AttributeValue.Builder`를 사용하여 `AttributeValue` 인스턴스를 생성할 수도 있습니다.

다음 코드 조각은 주석 줄 2 뒤에 두 개의 항목이 있는 맵을 보여줍니다. `expression()` 메서드에 전달된 문자열(주석 줄 1 뒤에 표시됨)에는 DynamoDB가 작업을 수행하기 전에 확인하는 자리 표시자가 포함되어 있습니다. 가격은 허용 가능한 속성 이름이므로 이 코드 조각에는 표현식 이름 맵이 포함되어 있지 않습니다.

```
    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();
```

DynamoDB 테이블의 속성 이름이 예약어이거나, 숫자로 시작하거나, 공백이 포함된 경우에는 Expression에 대해 표현식 이름 맵이 필요합니다.

예를 들어 이전 코드 예제의 속성 이름이 *price*가 *1price*인 경우 다음의 예제와 같이 예제를 수정해야 합니다.

```
        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();
```

표현식 이름의 자리 표시자는 파운드 기호(#)로 시작합니다. 표현식 이름 맵의 항목은 자리 표시자를 키로 사용하고 속성 이름을 값으로 사용합니다. 맵은 expressionNames() 메서드를 사용하여 표현식 빌더에 추가됩니다. DynamoDB는 작업을 수행하기 전에 속성 이름을 확인합니다.

문자열 표현식에 함수를 사용하는 경우에는 표현식 값이 필요하지 않습니다. 표현식 함수의 예는 attribute_exists(*<attribute_name>*)과 같습니다.

다음 예제는 DynamoDB 함수를 사용하여 Expression를 빌드합니다. 이 예제의 표현식 문자열은 자리 표시자를 사용하지 않습니다. 이 표현식은 movie 속성 값이 데이터 개체의 movie 속성과 같은 항목이 데이터베이스에 이미 존재하는지 확인하는 putItem 작업에 사용할 수 있습니다.

```
Expression exp = Expression.builder().expression("attribute_not_exists
 (movie)").build();
```

DynamoDB 개발자 안내서에는 DynamoDB와 함께 사용되는 하위 수준 표현식에 대한 전체 정보가 포함되어 있습니다.

조건 표현식 및 조건문

putItem(), updateItem() 및 deleteItem() 메서드를 사용할 때와 트랜잭션 및 배치 작업을 사용할 때는 Expression 객체를 사용하여 DynamoDB가 작업을 진행하기 위해 충족해야 하는 조건을 지정합니다. 이러한 식을 조건식이라고 합니다. 예제는 이 가이드에 나와 있는 트랜잭션 예제의 addDeleteItem() 메서드(주석 줄 1 뒤)에 사용된 조건식을 참조하세요.

query() 메서드를 사용하여 작업할 경우 조건은 QueryConditional로 표현됩니다. QueryConditional 클래스에는 DynamoDB에서 읽을 항목을 결정하는 기준을 작성하는 데 도움이 되는 몇 가지 정적 편의 메서드가 있습니다.

QueryConditionals의 예제는 이 가이드 the section called “Query 메서드 예제” 단원의 첫 번째 코드 예제를 참조하세요.

필터 표현식

필터 표현식은 스캔 및 쿼리 작업에서 반환되는 항목을 필터링하는 데 사용됩니다.

데이터베이스에서 모든 데이터를 읽은 후 필터 표현식이 적용되므로 읽기 비용은 필터가 없는 것과 동일합니다. Amazon DynamoDB 개발자 안내서에는 쿼리 및 스캔 작업 모두에 필터 표현식을 사용하는 방법에 대한 자세한 정보가 있습니다.

다음 예제는 스캔 요청에 추가된 필터 표현식을 보여줍니다. 기준에 따라 반품되는 품목은 가격이 8.00에서 80.00 사이인 품목으로 제한됩니다.

```
        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();
```

업데이트 표현식

DynamoDB 향상된 클라이언트의 updateItem() 메서드는 DynamoDB의 항목을 업데이트하는 표준 방법을 제공합니다. 하지만 더 많은 기능이 필요한 경우 UpdateExpressions는 DynamoDB 업데이트 표현식 구문을 형식에 구애받지 않고 표현할 수 있습니다. 예를 들어 DynamoDB에서 항목을 먼저 읽지 않고 값을 늘리거나 개별 구성원을 목록에 추가하는 데 UpdateExpressions을 사용할 수 있습니다. 업데이트 표현식은 updateItem() 메서드의 사용자 지정 확장에서 사용할 수 있습니다.

업데이트 표현식을 사용하는 예제는 이 가이드의 사용자 지정 확장 예제를 참조하세요.

업데이트 표현식에 대한 자세한 내용은 Amazon DynamoDB 개발자 안내서를 참조하세요.

페이지 매김된 결과 작업: 스캔 및 쿼리

DynamoDB 향상된 클라이언트 API의 `scan`, `query` 및 `batch` 메서드는 하나 이상의 페이지가 포함된 응답을 반환합니다. 페이지에는 하나 이상의 항목이 포함되어 있습니다. 코드는 페이지별로 응답을 처리하거나 개별 항목을 처리할 수 있습니다.

동기 `DynamoDbEnhancedClient` 클라이언트가 반환한 페이지 단위 응답은 PageIterable 객체를 반환하고, 비동기 `DynamoDbEnhancedAsyncClient`에서 반환한 응답은 PagePublisher 객체를 반환합니다.

이 단원에서는 페이지를 매긴 결과 처리를 살펴보고 스캔 및 쿼리 API를 사용하는 예제를 제공합니다.

테이블 스캔

SDK의 `scan` 메서드는 동일한 이름의 DynamoDB 작업에 해당합니다. DynamoDB 향상된 클라이언트 API는 동일한 옵션을 제공하지만 익숙한 객체 모델을 사용하고 페이지 매김을 자동으로 처리합니다.

먼저 동기 매핑 클래스 DynamoDBTable의 `scan` 메서드를 살펴보면서 `PageIterable` 인터페이스를 살펴봅니다.

동기식 API를 사용

다음 예제는 표현식을 사용하여 반환되는 항목을 필터링하는 `scan` 메서드를 보여줍니다. ProductCatalog는 이전에 표시된 모델 객체입니다.

주석 줄 2 뒤에 표시되는 필터링 표현식은 반품되는 품목을 가격 값이 8.00에서 80.00 사이인 `ProductCatalog` 항목만 포함하도록 제한합니다.

또한 이 예에서는 주석 줄 1 다음에 표시된 `attributesToProject` 방법을 사용하여 `isbn` 값을 제외합니다.

주석 줄 3 이후에는 `scan` 메서드가 `PageIterable` 객체, `pagedResults`를 반환합니다. `PageIterable`의 `stream` 메서드는 페이지를 처리하는 데 사용할 수 있는 `java.util.Stream` 객체를 반환합니다. 이 예제는 페이지 수를 세고 기록합니다.

이 예제는 주석 줄 4부터 시작하여 `ProductCatalog` 항목 액세스의 두 가지 변형을 보여줍니다. 주석 줄 4a 이후 버전은 각 페이지를 스트리밍하고 각 페이지의 항목을 정렬하고 기록합니다. 주석 줄 4b 이후의 버전은 페이지 이터레이션을 건너뛰고 항목에 직접 액세스합니다.

PageIterable 인터페이스는 2개의 상위 인터페이스 java.lang.Iterable 및 SdkIterable로 결과를 처리하는 다양한 방법을 제공합니다. Iterable은 forEach, iterator 및 spliterator 메서드를 가져오고 SdkIterable은 stream 메서드를 가져옵니다.

```
    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())
                );
    }
```

비동기 API 사용

비동기 scan 메서드는 결과를 PagePublisher 객체로 반환합니다. PagePublisher 인터페이스에는 응답 페이지를 처리하는 데 사용할 수 있는 두 가지 subscribe 메서드가 있습니다. 한 가지 subscribe 메서드는 org.reactivestreams.Publisher 상위 인터페이스에서 가져온 것입니다. 이 첫 번째 옵션을 사용하여 페이지를 처리하려면 subscribe 메서드에 Subscriber 인스턴스를 전달하세요. 다음 예제는 subscribe 메서드 사용을 보여 줍니다.

두 번째 subscribe 메서드는 SdkPublisher 인터페이스에서 가져온 것입니다. subscribe의 이 버전에서는 Subscriber가 아닌 Consumer를 받아들입니다. 이 subscribe 메서드 변형은 다음 두 번째 예제에 나와 있습니다.

다음 예제는 이전 예제와 동일한 필터 표현식을 사용하는 scan 메서드의 비동기 버전을 보여줍니다.

주석 줄 3번 다음에 DynamoDbAsyncTable.scan는 PagePublisher 객체를 반환합니다. 다음 줄에서 코드는 org.reactivestreams.Subscriber 인터페이스의 인스턴스를 만들고, ProductCatalogSubscriber는 주석 줄 4 뒤 PagePublisher를 구독합니다.

Subscriber 객체는 ProductCatalogSubscriber 클래스 예제의 주석 줄 8 다음에 있는 onNext 메서드의 각 페이지에서 ProductCatalog 항목을 수집합니다. 항목은 전용 List 변수에 저장되며 ProductCatalogSubscriber.getSubscribedItems() 메서드를 사용하여 호출 코드에서 액세스할 수 있습니다. 이는 주석 줄 5 이후에 호출됩니다.

목록이 검색되면 코드는 모든 ProductCatalog 항목을 가격별로 정렬하고 각 항목을 기록합니다.

ProductCatalogSubscriber 클래스의 CountDownLatch는 모든 항목이 목록에 추가될 때까지 호출 스레드를 차단한 다음 주석 줄 5 이후에 계속합니다.

```
    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;
        }
    }
```

다음 코드 조각 예제는 주석 줄 6을 다음에 Consumer를 받아들이는 PagePublisher.subscribe 메서드의 버전을 사용합니다. Java 람다 매개변수는 각 항목을 추가로 처리하는 페이지를 사용합니다. 이 예에서는 각 페이지가 처리되고 각 페이지의 항목이 정렬된 후 기록됩니다.

```
        // 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.
```

PagePublisher의 items 메서드는 모델 인스턴스를 언래핑하여 코드가 항목을 직접 처리할 수 있도록 합니다. 이 접근 방법은 다음 코드 조각에 나와 있습니다.

```
        // 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.
```

테이블 쿼리

DynamoDB 향상된 클라이언트를 사용하여 테이블을 쿼리하고 특정 기준과 일치하는 여러 항목을 검색할 수 있습니다. query() 메서드는 데이터 클래스에 정의된 @DynamoDbPartitionKey 및 @DynamoDbSortKey 주석(선택 사항)을 사용하여 프라이머리 키 값을 기반으로 항목을 찾습니다.

query() 메서드에는 파티션 키 값이 필요하며 경우에 따라 정렬 키 조건을 수락하여 결과를 추가로 구체화합니다. scan API와 마찬가지로 쿼리는 동기식 호출의 PageIterable을 반환하고 비동기식 호출의 PagePublisher를 반환합니다.

Query 메서드 예제

다음 query() 메서드 코드 예제에서는 MovieActor 클래스를 사용합니다. 데이터 클래스는 파티션 키의 **movie** 속성과 정렬 키의 **actor** 속성으로 구성된 복합 프라이머리 키를 정의합니다.

MovieActor 클래스

```
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);
        }
    }
}
```

다음 항목에 대한 쿼리를 따르는 코드 예제.

MovieActor 테이블 내 항목

```
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'}
```

다음 코드는 keyEqual(주석 줄 1 이후) 및 sortGreaterThanOrEqualTo(주석 줄 1a 이후)의 2가지 QueryConditional 인스턴스를 정의합니다.

파티션 키로 항목 쿼리

keyEqual 인스턴스는 파티션 키 값이 **movie01**인 항목과 일치합니다.

또한 이 예제는 주석 줄 2 이후 **actingschoolname**이 없는 항목을 필터링하는 필터 표현식을 정의합니다.

QueryEnhancedRequest는 쿼리에 대한 키 조건과 필터 표현식을 통합합니다.

```
    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 using the "keyEqual" conditional and filter expression.
        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.
                );
```

Example - keyEqual 쿼리 조건부를 사용한 출력

다음은 메서드 실행 결과입니다. 출력에는 movie01의 `movieName` 값이 있는 항목이 표시되고 **`null`**과 같은 `actingSchoolName`이 있는 항목은 표시되지 않습니다.

```
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'}
```

파티션 키 및 정렬 키로 항목 쿼리

`sortGreaterThanOrEqualTo QueryConditional`은 actor2보다 크거나 같은 값에 정렬 키 조건을 추가하여 파티션 키 일치(movie01)를 구체화합니다.

`sort`로 시작하는`QueryConditional 메서드`는 정렬 키 값을 기반으로 비교를 통해 쿼리를 일치시키고 더 구체화해야 합니다. 메서드 이름의 `Sort`는 결과가 정렬되었음을 의미하지 않지만 비교 시 정렬 키 값이 사용됩니다.

다음 코드 조각에서는 주석 줄 3 이후, 앞서 표시된 쿼리 요청을 변경합니다. 이 코드 조각은 'keyEqual' 쿼리 조건을 주석 줄 1a 이후 정의된 'sortGreaterThanOrEqualTo' 쿼리 조건으로 대체합니다. 다음 코드는 또한 필터 표현식을 제거합니다.

```
        QueryEnhancedRequest tableQuery = QueryEnhancedRequest.builder()
                .queryConditional(sortGreaterThanOrEqualTo).build();
```

Example - sortGreaterThanOrEqualTo 쿼리 조건부를 사용한 출력

다음 출력은 쿼리 결과를 표시합니다. 쿼리는 `movieName` 값이 movie01과 같은 항목을 반환하고 actor2보다 크거나 같은 `actorName` 값을 가진 항목만 반환합니다. 필터가 제거되었으므로 쿼리는 `actingSchoolName` 속성 값이 없는 항목을 반환합니다.

```
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'}
```

배치 작업 수행

DynamoDB 향상된 클라이언트 API는 batchGetItem() 및 batchWriteItem()라는 두 가지 배치 메서드를 제공합니다.

batchGetItem() 예제

이 DynamoDbEnhancedClient.batchGetItem() 메서드를 사용하면 전체 요청 한 번으로 여러 테이블에서 최대 100개의 개별 항목을 검색할 수 있습니다. 다음 예제에서는 이전에 표시된 Customer 및 MovieActor 데이터 클래스를 사용합니다.

1행과 2행 뒤에 오는 예제에서는 ReadBatch 객체를 빌드하고 나중에 이 객체를 batchGetItem() 메서드에 주석 줄 3 다음에 파라미터로 추가합니다.

주석 줄 1 뒤의 코드는 Customer 테이블에서 읽을 배치를 빌드합니다. 주석 줄 1a 뒤의 코드는 프라이머리 키 값과 정렬 키 값을 사용하여 읽을 항목을 지정하는 GetItemEnhancedRequest 빌더를 사용하는 방법을 보여줍니다. 데이터 클래스에 복합 키가 있는 경우 파티션 키 값과 정렬 키 값을 모두 제공해야 합니다.

항목을 요청하기 위해 키 값을 지정하는 것과 달리 주석 줄 1b 다음에 표시된 대로 데이터 클래스를 사용하여 항목을 요청할 수 있습니다. SDK는 요청을 제출하기 전에 백그라운드에서 키 값을 추출합니다.

2a 이후의 두 명령문에서 볼 수 있듯이 키 기반 접근 방식을 사용하여 항목을 지정하는 경우 DynamoDB가 매우 일관된 읽기를 수행하도록 지정할 수도 있습니다. consistentRead() 메서드를 사용하는 경우 동일한 테이블에 대해 요청된 모든 항목에 이 메서드를 사용해야 합니다.

DynamoDB에서 찾은 항목을 검색하려면 주석 4줄 뒤에 표시된 resultsForTable() 메서드를 사용하세요. 요청에서 읽은 각 테이블의 메서드를 호출합니다. resultsForTable()는 임의의 java.util.List 메서드를 사용하여 처리할 수 있는 검색된 항목 목록을 반환합니다. 이 예제는 각 항목을 기록합니다.

DynamoDB에서 처리하지 않은 항목을 검색하려면 주석 5줄 다음에 있는 접근 방식을 사용하세요. `BatchGetResultPage` 클래스에는 처리되지 않은 각 키에 액세스할 수 있는 `unprocessedKeysForTable()` 메서드가 있습니다. BatchGetItem API 참조에는 처리되지 않은 항목이 발생하는 상황에 대한 자세한 정보가 있습니다.

```
    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 value and sort key value 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(movieActorTable).forEach(key ->
 logger.info("Unprocessed item key: " + key.toString()));
        });
    }
```

예제 코드를 실행하기 전에 두 테이블에 다음 항목이 있다고 가정해 보세요.

테이블 내 항목

```
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'}
```

다음 출력은 주석 라인 4 이후에 반환되고 기록된 항목을 보여줍니다.

```
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() 예제

이 메서드는 하나 이상의 테이블에 여러 항목을 추가하거나 삭제합니다. 요청에 최대 25개의 개별 넣기 또는 삭제 작업을 지정할 수 있습니다. 다음 예제에서는 이전에 표시된 ProductCatalog 및 MovieActor 데이터 클래스를 사용합니다.

WriteBatch 객체는 주석 라인 1과 2 다음에 빌드됩니다. ProductCatalog 테이블의 경우 코드는 항목 하나를 추가하고 항목 하나를 삭제합니다. 주석 줄 2 뒤에 있는 MovieActor 테이블의 경우 코드는 항목 두 개를 넣고 한 개를 삭제합니다.

batchWriteItem 메서드는 주석 줄 3 이후에 호출됩니다. builder 파라미터는 각 테이블에 대한 일괄 요청을 제공합니다.

반환된 BatchWriteResult 객체는 처리되지 않은 요청을 볼 수 있는 각 작업에 대해 별도의 메서드를 제공합니다. 주석 줄 4a 뒤의 코드는 처리되지 않은 삭제 요청에 대한 키를 제공하고 주석 줄 4b 뒤의 코드는 처리되지 않은 put 항목을 제공합니다.

```
    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()));
        }
    }
```

다음 도우미 메서드는 업로드 및 삭제 작업을 위한 모델 객체를 제공합니다.

도우미 메서드

```
    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;
    }
```

예제 코드를 실행하기 전에 테이블에 다음 항목이 포함되어 있다고 가정해 보겠습니다.

```
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}
```

예제 코드가 끝나면 테이블에는 다음 항목이 포함됩니다.

```
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}
```

`MovieActor` 테이블에서 `Blue Jasmine` 영화 항목이 `getMovieActorBlanchettPartial()` 도우미 메서드를 통해 획득한 put 요청에 사용된 항목으로 대체되었음을 알 수 있습니다. 데이터 bean 속성 값이 제공되지 않은 경우 데이터베이스의 값이 제거됩니다. 이것이 `Blue Jasmine` 영화 항목의 결과 `actingSchoolName`가 null이 되는 이유입니다.

> **Note**
>
> API 설명서에는 조건식을 사용할 수 있고 사용된 용량 및 컬렉션 지표를 개별 PUT 및 DELETE 요청과 함께 반환할 수 있다고 나와 있지만 일괄 쓰기 시나리오에서는 그렇지 않습니다. 일괄 작업의 성능을 향상시키기 위해 이러한 개별 옵션은 무시됩니다.

트랜잭션 작업 수행

DynamoDB 향상된 클라이언트 API는 `transactGetItems()` 및 `transactWriteItems()` 메서드를 제공합니다. Java용 SDK의 트랜잭션 방법은 DynamoDB 테이블에 ACID(원자성, 일관성, 격리 및 내구성)를 제공하여 애플리케이션에서 데이터 정확성을 유지하는 데 도움이 됩니다.

`transactGetItems()` 예제

`transactGetItems()` 메서드는 항목에 대한 개별 요청을 최대 100개까지 수락합니다. 단일 아토믹 트랜잭션으로 모든 항목을 읽습니다. Amazon DynamoDB 개발자 안내서에는 `transactGetItems()` 메서드 실패를 유발하는 조건에 대한 정보와 `transactGetItem()` 호출 시 사용되는 격리 수준에 대한 정보가 있습니다.

다음 예제의 주석 줄 1 다음에, 코드는 `builder` 매개변수를 사용하여 `transactGetItems()` 메서드를 호출합니다. SDK가 최종 요청을 생성하는 데 사용할 키 값이 포함된 데이터 객체를 사용하여 빌더 `addGetItem()`를 세 번 호출합니다.

요청은 주석 줄 2 다음에 `Document` 객체 목록을 반환합니다. 반환되는 문서 목록에는 요청된 순서와 동일한 순서로 항목 데이터의 null이 아닌 `Document` 인스턴스가 포함되어 있습니다. `Document.getItem(MappedTableResource<T> mappedTableResource)` 메서드는 항목 데이

터가 반환된 경우 형식화되지 않은 Document 객체를 형식이 지정된 Java 객체로 변환하고, 그렇지 않으면 null을 반환합니다.

```
   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());
       }
   }
```

코드 예제가 실행되기 전에 DynamoDB 테이블에는 다음 항목이 포함되어 있습니다.

```
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'}
```

다음 출력이 반환됩니다. 항목을 요청했지만 찾을 수 없는 경우 이름이 `Blue Jasmine`로 지정된 영화에 대한 요청의 경우와 마찬가지로 항목이 반환되지 않습니다.

```
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()` 예제

`transactWriteItems()`는 여러 테이블에 걸친 단일 아토믹 트랜잭션으로 최대 100개의 추가, 업데이트 또는 삭제 작업을 허용합니다. Amazon DynamoDB 개발자 안내서에는 기본 DynamoDB 서비스 작업의 제한 및 장애 조건에 대한 세부 정보가 포함되어 있습니다.

기본 예제

다음 예제에서는 2개의 테이블에 대해 4개의 작업이 요청됩니다. 해당 모델 클래스 `ProductCatalog` 및 `MovieActor`는 이전에 표시된 바 있습니다.

세 가지 가능한 작업(put, update, delete)은 각각 전용 요청 매개변수를 사용하여 세부 정보를 지정합니다.

주석 줄 1 뒤의 코드는 `addPutItem()` 메서드의 단순한 변형을 보여줍니다. 메서드는 입력할 `MappedTableResource` 개체와 데이터 객체 인스턴스를 받아들입니다. 주석 줄 2 뒤의 명령문은 `TransactPutItemEnhancedRequest` 인스턴스를 허용하는 변형을 보여줍니다. 이 변형을 사용하면 요청에 조건 표현식과 같은 더 많은 옵션을 추가할 수 있습니다. 다음 예제에서는 개별 작업에 대한 조건식을 보여줍니다.

주석 줄 3 다음에 업데이트 작업이 요청됩니다. `TransactUpdateItemEnhancedRequest`에는 SDK가 모델 객체의 `null` 값으로 수행하는 작업을 구성할 수 있는 `ignoreNulls()` 메서드가 있습니다. `ignoreNulls()` 메서드가 true를 반환하는 경우 SDK는 `null`인 데이터 객체 속성에 대한 테이블의 속성 값을 제거하지 않습니다. `ignoreNulls()` 메서드가 false를 반환하면 SDK는 DynamoDB 서비스에 테이블의 항목에서 속성을 제거하도록 요청합니다. `ignoreNulls`의 기본값은 false입니다.

주석 줄 4 다음의 명령문은 데이터 객체를 취하는 삭제 요청의 변형을 보여줍니다. 향상된 클라이언트는 최종 요청을 발송하기 전에 키 값을 추출합니다.

```
    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())
        );
    }
```

다음 도우미 메서드는 add*Item 매개 변수에 대한 데이터 개체를 제공합니다.

도우미 메서드

```
    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;
    }
```

코드 예제가 실행되기 전에 DynamoDB 테이블에는 다음 항목이 포함되어 있습니다.

```
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'}
```

코드 실행이 완료된 후 테이블에 다음 항목이 표시됩니다.

```
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'}
```

2행 항목은 삭제되었으며, 3행과 5행에는 추가된 항목이 표시됩니다. 4행에는 라인 1의 업데이트 내용이 표시됩니다. `price` 값은 항목에서 변경된 유일한 값입니다. `ignoreNulls()`가 false를 반환했다면 4행은 다음 줄처럼 보일 것입니다.

```
ProductCatalog{id=4, title='Title 1', isbn='null', authors=null, price=40.0}
```

조건 검사 예제

다음 예에서는 조건 검사의 사용을 보여줍니다. 조건 검사는 항목이 있는지 확인하거나 데이터베이스의 항목의 특정 속성에 대한 조건을 검사하는 데 사용됩니다. 조건 확인에서 확인한 항목은 해당 거래의 다른 작업에 사용할 수 없습니다.

> **Note**
>
> 동일한 트랜잭션 내에서 동일한 항목을 여러 작업의 대상으로 지정할 수 없습니다. 예를 들어 동일한 트랜잭션에서 조건 확인과 동일한 항목 업데이트를 동시에 수행할 수 없습니다.

이 예에서는 트랜잭션 쓰기 항목 요청의 각 작업 유형 중 하나를 보여줍니다. `addConditionCheck()` 메서드는 주석 줄 2 다음에 `conditionExpression` 매개 변수가 `false`로 평가될 경우 트랜잭션을 실패시키는 조건을 제공합니다. Helper 메서드 블록에 표시된 메서드에서 반환되는 조건 표현식은 영화 `Sophie's Choice`의 수상 연도가 1982과 같지 않은지 확인합니다. 값이 맞으면 표현식이 `false`까지 평가되고 트랜잭션이 실패합니다.

이 가이드에서는 다른 항목에서 표현식에 대해 자세히 설명합니다.

```
    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());
            });
        }
    }
```

이전 코드 예제에서는 다음과 같은 도우미 메서드를 사용했습니다.

도우미 메서드

```
    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;
    }
```

코드 예제가 실행되기 전에 DynamoDB 테이블에는 다음 항목이 포함되어 있습니다.

```
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'}
```

코드 실행이 완료된 후 테이블에 다음 항목이 표시됩니다.

```
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'}
```

트랜잭션이 실패했기 때문에 테이블의 항목은 변경되지 않은 상태로 유지됩니다. 동영상 Sophie's Choice의 actingYear 값은 1982으로, transactWriteItem() 메서드가 호출되기 전 테이블 항목의 2행에 표시된 것과 같습니다.

트랜잭션에 대한 취소 정보를 캡처하려면 transactWriteItems() 메서드 호출을 try 블록으로 묶고 TransactionCanceledException를 catch합니다. 예제의 주석 줄 4 다음에 코드가 각 CancellationReason 객체를 기록합니다. 예제의 주석 줄 3 뒤에 오는 코드에서는 트랜잭션 실패를 초래한 항목에 대해 값을 반환하도록 지정하기 때문에 로그에는 Sophie's Choice 영화 항목에 대한 원시 데이터베이스 값이 표시됩니다.

```
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.)
```

단일 작업 조건 예제

다음 예제는 트랜잭션 요청의 단일 작업에 대한 조건 사용을 보여줍니다. 주석 라인 1 이후의 삭제 작업에는 데이터베이스에 대해 작업의 대상 항목 값을 확인하는 조건이 포함되어 있습니다. 이 예제에서 주석 행 2 다음에 도우미 메서드를 사용하여 생성된 조건식은 영화의 개봉 연도가 2013년이 아닌 경우 데이터베이스에서 항목을 삭제해야 함을 지정합니다.

표현식은 이 가이드의 뒷부분에서 설명합니다.

```
   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();
   }
```

이전 코드 예제에서는 다음과 같은 도우미 메서드를 사용했습니다.

도우미 메서드

```
   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;
    }
```

코드 예제가 실행되기 전에 DynamoDB 테이블에는 다음 항목이 포함되어 있습니다.

```
1 | ProductCatalog{id=4, title='Title 1', isbn='orig_isbn', authors=[b, g], price=10}
2 | MovieActor{movieName='Blue Jasmine', actorName='Cate Blanchett', actingAward='Best
 Actress', actingYear=2013, actingSchoolName='National Institute of Dramatic Art'}
```

코드 실행이 완료된 후 테이블에 다음 항목이 표시됩니다.

```
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'}
```

트랜잭션이 실패했기 때문에 테이블의 항목은 변경되지 않은 상태로 유지됩니다. 영화 `Blue Jasmine`의 `actingYear` 값은 코드 예제가 실행되기 전 항목 목록의 2행에 표시된 것과 같은 `2013`입니다.

다음 줄이 콘솔에 기록됩니다.

```
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)
```

보조 인덱스 사용

보조 인덱스는 쿼리 및 스캔 작업에 사용하는 대체 키를 정의하여 데이터 액세스를 향상시킵니다. GSI(글로벌 보조 인덱스)에는 기본 테이블의 파티션 키와 정렬 키가 다를 수 있습니다. 반대로 LSI(로컬 보조 인덱스)는 기본 인덱스의 파티션 키를 사용합니다.

보조 인덱스 주석으로 데이터 클래스에 주석 달기

보조 인덱스에 속하는 속성에는 `@DynamoDbSecondaryPartitionKey` 또는 `@DynamoDbSecondarySortKey` 주석이 필요합니다.

다음 클래스는 두 인덱스에 대한 주석을 보여줍니다. SubjectLastPostedDateIndex라는 GSI는 파티션 키에 `Subject` 속성을 사용하고 정렬 키에는 `LastPostedDateTime` 속성을 사용합니다. ForumLastPostedDateIndex라는 이름의 LSI는 `ForumName`를 파티션 키로, `LastPostedDateTime`를 정렬 키로 사용합니다.

`Subject` 속성은 이중 역할을 한다는 점에 유의하세요. 기본 키의 정렬 키이자 SubjectLastPostedDateIndex라는 GSI의 파티션 키입니다.

`MessageThread` 클래스

이 `MessageThread` 클래스는 Amazon DynamoDB 개발자 안내서의 예제 스레드 테이블의 데이터 클래스로 사용하기에 적합합니다.

가져오기

```
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 +
                '}';
    }
}
```

인덱스 만들기

Java용 SDK 버전 2.20.86부터 이 `createTable()` 메서드는 데이터 클래스 주석에서 보조 인덱스를 자동으로 생성합니다. 기본적으로 기본 테이블의 모든 속성이 인덱스에 복사되며 프로비저닝된 처리량 값은 20 읽기 용량 단위 및 20 쓰기 용량 단위입니다.

단, SDK 2.20.86 이전 버전을 사용하는 경우에는 다음 예와 같이 테이블과 함께 인덱스를 빌드해야 합니다. 이 예제에서는 `Thread` 테이블의 인덱스 두 개를 빌드합니다. 빌더 매개 변수에는 주석 줄 1과 2 다음에 표시된 것처럼 두 가지 유형의 인덱스를 모두 구성하는 메서드가 있습니다. 인덱스 빌더의 `indexName()` 메서드를 사용하여 데이터 클래스 주석에 지정된 인덱스 이름을 의도한 인덱스 유형과 연결할 수 있습니다.

이 코드는 모든 테이블 속성이 주석 라인 3과 4 다음에 있는 두 인덱스 모두에 포함되도록 구성합니다. 속성 프로젝션에 대한 자세한 내용은 Amazon DynamoDB 개발자 안내서에서 확인할 수 있습니다.

```
   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))
               )
       );
```

인덱스를 사용하여 쿼리

다음 예시에서는 로컬 보조 인덱스 ForumLastPostedDateIndex를 쿼리합니다.

주석 줄 2에 이어 DynamoDbIndex.query() 메서드를 호출할 때 필요한 QueryConditional 객체를 생성합니다.

인덱스 이름을 전달하여 주석 줄 3 다음에 쿼리하려는 인덱스에 대한 참조를 얻습니다. 주석 줄 4에 이어 `QueryConditional` 객체를 전달하는 인덱스에서 `query()` 메서드를 호출합니다.

또한 주석 줄 5 다음에 표시된 대로 세 가지 속성 값을 반환하도록 쿼리를 구성합니다. `attributesToProject()`가 호출되지 않은 경우 쿼리는 모든 속성 값을 반환합니다. 지정된 속성 이름은 소문자로 시작하는 것을 알 수 있습니다. 이러한 속성 이름은 테이블에 사용된 이름과 일치하지만 반드시 데이터 클래스의 속성 이름과 일치할 필요는 없습니다.

주석 줄 6 다음에 결과를 반복하고 쿼리에서 반환된 각 항목을 기록하고 목록에 저장하여 호출자에게 반환합니다.

```
public class IndexScanExamples {
    private static Logger logger = LoggerFactory.getLogger(IndexScanExamples.class);
```

```
   public static List<MessageThread> queryUsingSecondaryIndices(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.
       final DynamoDbIndex<MessageThread> forumLastPostedDateIndex =
threadTable.index("ForumLastPostedDateIndex");

       // 4. Perform the query 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 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;
   }
```

쿼리가 실행되기 전의 데이터베이스에는 다음과 같은 항목이 있습니다.

```
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}
```

1행과 6행의 로깅 명령문을 실행하면 다음과 같은 콘솔 출력이 나타납니다.

```
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}
```

쿼리가 Forum02의 `forumName` 값과 2023.03.31보다 크거나 같은 `lastPostedDateTime`을 반환했습니다. 인덱스에 `message` 속성 값이 있더라도 결과에는 빈 문자열이 있는 `message` 값이 표시됩니다. 이는 주석 줄 5 이후에 코드로 메시지 속성이 프로젝션되지 않았기 때문입니다.

고급 매핑 기능 사용

DynamoDB 향상된 클라이언트 API의 고급 테이블 스키마 기능에 대해 알아보세요.

테이블 스키마 유형 이해

`TableSchema`는 DynamoDB 향상된 클라이언트 API의 매핑 기능에 대한 인터페이스입니다. AttributeValues 맵에 데이터 객체를 매핑하거나 이 맵에서 데이터 객체를 매핑할 수 있습니다. `TableSchema` 객체는 매핑하려는 테이블의 구조를 알아야 합니다. 이 구조 정보는 `TableMetadata` 객체에 저장됩니다.

향상된 클라이언트 API에는 `TableSchema`의 여러 구현이 있습니다.

주석이 달린 클래스에서 생성된 테이블 스키마

주석이 달린 클래스로 `TableSchema`를 빌드하는 작업은 비용이 다소 많이 들기 때문에 애플리케이션을 시작할 때 한 번 수행하는 것이 좋습니다.

BeanTableSchema

이 구현은 Bean 클래스의 속성 및 주석을 기반으로 빌드됩니다. 이 접근 방식의 예제는 시작하기 단원에 나와 있습니다.

> **Note**
>
> `BeanTableSchema`가 예상대로 작동하지 않는 경우 `software.amazon.awssdk.enhanced.dynamodb.beans`에 대한 디버그 로깅을 활성화하세요.

ImmutableTableSchema

이 구현은 변경할 수 없는 데이터 클래스를 기반으로 구축되었습니다. 이 접근은 ??? 단원에서 설명합니다.

빌더로 생성된 테이블 스키마

다음 `TableSchema`은 빌더를 사용하여 코드로 빌드됩니다. 이 접근 방식은 주석이 달린 데이터 클래스를 사용하는 접근 방식보다 비용이 저렴합니다. 빌더 접근 방식은 주석을 사용하지 않으므로 JavaBean 이름 지정 표준이 필요하지 않습니다.

StaticTableSchema

이 구현은 변경 가능한 데이터 클래스용으로 구축되었습니다. 이 가이드의 시작하기 단원에서는 빌더를 사용하여 StaticTableSchema를 생성하는 방법을 보여 주었습니다.

StaticImmutableTableSchema

StaticTableSchema를 빌드하는 방법과 마찬가지로 변경할 수 없는 데이터 클래스와 함께 사용할 빌더를 사용하여 이러한 유형의 TableSchema 구현을 생성합니다.

고정 스키마가 없는 데이터에 대한 테이블 스키마

DocumentTableSchema

TableSchema의 다른 구현과 달리 DocumentTableSchema 인스턴스의 속성은 정의하지 않습니다. 일반적으로 기본 키와 특성 변환기 공급자만 지정합니다. EnhancedDocument 인스턴스는 개별 요소 또는 JSON 문자열에서 빌드한 속성을 제공합니다.

속성을 명시적으로 포함하거나 제외

DynamoDB 향상된 클라이언트 API는 데이터 클래스 속성이 테이블의 속성이 되지 않도록 제외하는 주석을 제공합니다. API를 사용하면 데이터 클래스 속성 이름과 다른 속성 이름을 사용할 수도 있습니다.

속성 제외

DynamoDB 테이블에 매핑해서는 안 되는 속성을 무시하려면 속성을 @DynamoDbIgnore 주석으로 표시하세요.

```
private String internalKey;

@DynamoDbIgnore
public String getInternalKey() { return this.internalKey; }
public void setInternalKey(String internalKey) { this.internalKey = internalKey;}
```

속성 포함

DynamoDB 테이블에서 사용되는 속성의 이름을 변경하려면 @DynamoDbAttribute 주석으로 표시하고 다른 이름을 제공하세요.

```
private String internalKey;
```

```
@DynamoDbAttribute("renamedInternalKey")
public String getInternalKey() { return this.internalKey; }
public void setInternalKey(String internalKey) { this.internalKey = internalKey;}
```

제어 속성 변환

기본적으로 테이블 스키마는 AttributeConverterProvider 인터페이스의 기본 구현을 통해 많은 공통 Java 유형에 대한 변환기를 제공합니다. 사용자 지정 AttributeConverterProvider 구현으로 전체 기본 동작을 변경할 수 있습니다. 단일 속성의 변환기를 변경할 수도 있습니다.

사용 가능한 컨버터 목록은 AttributeConverter 인터페이스 Java 문서를 참조하세요.

사용자 정의 속성 변환기 공급자 제공

@DynamoDbBean (converterProviders = {…})주석을 통해 정렬된 단일 AttributeConverterProvider 또는 순서로 정리된 AttributeConverterProvider의 체인을 제공할 수 있습니다. 모든 사용자 지정 AttributeConverterProvider은 AttributeConverterProvider 인터페이스를 확장해야 합니다.

자체 특성 변환기 공급자 체인을 제공하는 경우 기본 변환기 공급자 DefaultAttributeConverterProvider가 재정의됨을 유의하세요. DefaultAttributeConverterProvider의 기능을 사용하려면 이 기능을 체인에 포함해야 합니다.

빈 배열 {}로 Bean에 주석을 달 수도 있습니다. 이렇게 하면 기본값을 포함하여 모든 특성 변환기 공급자의 사용이 비활성화됩니다. 이 경우 매핑할 모든 속성에는 고유한 속성 변환기가 있어야 합니다.

다음 코드 조각은 단일 변환기 제공자를 보여줍니다.

```
@DynamoDbBean(converterProviders = ConverterProvider1.class)
public class Customer {

}
```

다음 코드 조각은 변환기 제공자 체인의 사용을 보여줍니다. SDK 기본값은 마지막에 제공되므로 우선순위가 가장 낮습니다.

```
@DynamoDbBean(converterProviders = {
   ConverterProvider1.class,
   ConverterProvider2.class,
   DefaultAttributeConverterProvider.class})
public class Customer {
```

```
}
```

정적 테이블 스키마 빌더에는 동일한 attributeConverterProviders() 방식으로 작동하는 메서드가 있습니다. 이는 다음 예제와 같습니다.

```
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();
```

단일 속성의 매핑 재정의

단일 속성이 매핑되는 방식을 재정의하려면 속성에 AttributeConverter을 제공하세요. 이 추가 기능은 테이블 스키마의 AttributeConverterProviders에서 제공하는 모든 변환기를 재정의합니다. 이렇게 하면 해당 속성에만 사용자 지정 변환기가 추가됩니다. 동일한 유형의 속성이라 할지라도, 다른 속성은 해당 속성에 대해 명시적으로 지정되지 않는 한 해당 변환기를 사용하지 않습니다.

@DynamoDbConvertedBy 주석은 다음 코드 조각과 같이 사용자 지정 AttributeConverter 클래스를 지정하는 데 사용됩니다.

```
@DynamoDbBean
public class Customer {
    private String name;

    @DynamoDbConvertedBy(CustomAttributeConverter.class)
    public String getName() { return this.name; }
    public void setName(String name) { this.name = name;}
}
```

정적 스키마용 빌더에는 동일한 속성 빌더 attributeConverter() 메서드가 있습니다. 이 메서드는 다음과 같이 AttributeConverter의 인스턴스를 사용합니다.

```
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();
```

예시

이 예제는 [java.net.HttpCookie](#) 객체의 속성 변환기를 제공하는 `AttributeConverterProvider` 구현을 보여줍니다.

다음 `SimpleUser` 클래스에는 `HttpCookie`의 인스턴스인 이름이 `lastUsedCookie`로 지정된 속성이 들어 있습니다.

`@DynamoDbBean` 주석의 매개 변수에는 변환기를 제공하는 두 `AttributeConverterProvider` 클래스가 나열되어 있습니다.

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();
```

다음 CookieConverterProvider 예제에서는 HttpCookeConverter의 인스턴스를 제공합니다.

```
   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);
       }
   }
```

전환 코드

다음 HttpCookieConverter 클래스의 transformFrom() 메서드에서 코드는 HttpCookie 인스턴스를 수신하여 속성으로 저장되는 DynamoDB 맵으로 변환합니다.

`transformTo()` 메서드는 DynamoDB 맵 파라미터를 수신한 다음 이름과 값이 필요한 생성자를 `HttpCookie` 호출합니다.

```
   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;
       }
   }
```

속성의 업데이트 동작 변경

업데이트 작업을 수행할 때 개별 속성의 업데이트 동작을 사용자 지정할 수 있습니다. DynamoDB 향상된 클라이언트 API의 업데이트 작업 예로는 updateItem() 및 transactWriteItems()가 있습니다.

예를 들어 생성된 타임스탬프를 레코드에 저장하고 싶다고 가정해 보겠습니다. 그러나 데이터베이스에 이미 속성에 대한 기존 값이 없는 경우에만 해당 값이 기록되기를 원합니다. 이 경우에는 `WRITE_IF_NOT_EXISTS` 업데이트 동작을 사용합니다.

다음 예제는 createdOn 속성에 동작을 추가하는 주석을 보여줍니다.

```
@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; }
}
```

주석 행 1 다음에 다음 예에 표시된 대로 정적 테이블 스키마를 빌드할 때 동일한 업데이트 동작을 선언할 수 있습니다.

```
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();
```

다른 클래스의 속성을 평면화

테이블 속성이 상속 또는 구성을 통해 여러 Java 클래스에 분산되어 있는 경우 DynamoDB 향상된 클라이언트 API는 속성을 하나의 클래스로 병합할 수 있도록 지원합니다.

상속 사용

클래스에서 상속을 사용하는 경우 다음 방법을 사용하여 계층 구조를 평면화하세요.

주석이 달린 빈을 사용

주석 접근 방식의 경우 두 클래스 모두 @DynamoDbBean 주석을 포함해야 하며 클래스에는 기본 키 주석이 하나 이상 있어야 합니다.

다음은 상속 관계가 있는 데이터 클래스의 예를 보여줍니다.

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

Lombok의 onMethod 옵션은 속성 기반 DynamoDB 주석(예: @DynamoDbPartitionKey)을 생성된 코드에 복사합니다.

```
@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;
}
```

정적 스키마 사용

정적 스키마 접근 방식의 경우 빌더의 `extend()` 메서드를 사용하여 부모 클래스의 속성을 자식 클래스로 축소합니다. 이는 다음 예에서 주석 라인 1 뒤에 표시됩니다.

```
       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();
```

이전 정적 스키마 예제는 다음의 데이터 클래스를 사용합니다. 정적 테이블 스키마를 작성할 때 매핑이 정의되므로 데이터 클래스에는 주석이 필요하지 않습니다.

데이터 클래스

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;
}
```

구성 사용

클래스에서 컴포지션을 사용하는 경우 다음 방법을 사용하여 계층 구조를 평면화하세요.

주석이 달린 빈을 사용

`@DynamoDbFlatten` 주석은 포함된 클래스를 평면화합니다.

다음 데이터 클래스 예제는 `@DynamoDbFlatten` 주석을 사용하여 포함된 `GenericRecord` 클래스의 모든 속성을 `Customer` 클래스에 효과적으로 추가합니다.

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;
}
```

평면화 주석을 사용하여 필요한 만큼 다양한 적격 클래스를 병합할 수 있습니다. 다음과 같은 제약이 적용됩니다.

- 병합한 후에는 모든 속성 이름이 고유해야 합니다.
- 파티션 키, 정렬 키 또는 테이블 이름이 두 개를 초과해서는 안 됩니다.

정적 스키마 사용

정적 테이블 스키마를 빌드할 때는 빌더의 `flatten()` 메서드를 사용하세요. 포함된 클래스를 식별하는 접근자 및 설정자 메서드도 제공합니다.

```
       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();
```

이전 정적 스키마 예제는 다음의 데이터 클래스를 사용합니다.

데이터 클래스

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;
}
```

빌더 패턴을 사용하여 필요한 만큼 다양한 적격 클래스를 평면화할 수 있습니다.

다른 코드에 미치는 영향

@DynamoDbFlatten 속성(또는 flatten( ) 빌더 메서드)을 사용하는 경우 DynamoDB의 항목은 구성된 객체의 각 속성에 대한 속성을 포함합니다. 또한 작성 객체의 속성도 포함됩니다.

반대로 작성된 클래스로 데이터 클래스에 주석을 달고 @DynamoDbFlatten를 사용하지 않는 경우 항목은 구성된 객체와 함께 단일 속성으로 저장됩니다.

예를 들어 평면화 구성 예제에 표시된 Customer 클래스를 record 속성을 평면화한 경우 및 사용하지 않은 경우와 비교해 보세요. 다음 표에 표시된 것처럼 JSON을 사용하여 차이점을 시각화할 수 있습니다.

평면화 사용	평면화 사용하지 않음
속성 3개	속성 2개

<pre>{
  "id": "1",
  "createdDate": "today",
  "name": "my name"
}</pre>
	<pre>{
	  "id": "1",
	  "record": {
	  "createdDate": "today",
	  "name": "my name"
	  }
	}</pre>

특정 속성을 찾을 것으로 예상하는 다른 코드가 DynamoDB 테이블에 액세스하는 경우 차이가 중요해집니다.

bean, 맵, 목록 및 세트로 구성된 속성으로 작업

아래 표시된 `Person` 클래스와 같은 bean 정의는 추가 속성이 있는 유형을 참조하는 속성(또는 속성)을 정의할 수 있습니다. 예를 들어 `Person` 클래스에서 `mainAddress`는 추가 값 속성을 정의하는 `Address` bean을 참조하는 속성입니다. `addresses`는 Java Map을 참조하며, 여기서 요소는 `Address` bean을 참조합니다. 이러한 복잡한 유형으로 DynamoDB의 컨텍스트에서 데이터 값에 사용하는 간단한 속성의 컨테이너를 떠올릴 수 있습니다.

DynamoDB는 맵, 목록 또는 bean과 같은 중첩 요소의 값 속성을 중첩 속성이라고 합니다. Amazon DynamoDB 개발자 안내서는 Java 맵, 목록 또는 bean의 저장된 형식을 문서 유형으로 참조합니다. Java에서 데이터 값에 사용하는 간단한 속성을 DynamoDB에서 스칼라 유형이라고 합니다. 동일한 유형의 여러 스칼라 요소를 포함한 집합을 집합 유형이라고 합니다.

DynamoDB 향상된 클라이언트 API는 저장 시 bean 속성을 DynamoDB 맵 문서 유형으로 변환한다는 것을 이해하는 것이 중요합니다.

`Person` 클래스

```
@DynamoDbBean
public class Person {
    private Integer id;
    private String firstName;
    private String lastName;
    private Integer age;
    private Address mainAddress;
    private Map<String, Address> addresses;
    private List<PhoneNumber> phoneNumbers;
    private Set<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 Address getMainAddress() {
        return mainAddress;
    }

    public void setMainAddress(Address mainAddress) {
        this.mainAddress = mainAddress;
    }

    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 Set<String> getHobbies() {
        return hobbies;
    }

    public void setHobbies(Set<String> hobbies) {
        this.hobbies = hobbies;
    }

    @Override
    public String toString() {
        return "Person{" +
               "addresses=" + addresses +
               ", id=" + id +
               ", firstName='" + firstName + '\'' +
               ", lastName='" + lastName + '\'' +
               ", age=" + age +
               ", mainAddress=" + mainAddress +
               ", phoneNumbers=" + phoneNumbers +
               ", hobbies=" + hobbies +
               '}';
    }
}
```

Address 클래스

```
@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` 클래스

```
@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 + '\'' +
                '}';
    }
}
```

복잡한 유형 저장

주석이 달린 데이터 클래스 사용

사용자 정의 클래스에 주석을 달기만 하면 사용자 정의 클래스의 중첩 속성을 저장할 수 있습니다. 이전에 표시된 Address 클래스와 PhoneNumber 클래스에는 @DynamoDbBean 주석만 있는 주석이 달려 있습니다. DynamoDB 향상된 클라이언트 API가 다음 코드 조각을 사용하여 클래스에 대한 Person 테이블 스키마를 구축하면 API는 Address 및 PhoneNumber 클래스의 사용을 발견하고 DynamoDB와 함께 작동하도록 해당 매핑을 구축합니다.

```
TableSchema<Person> personTableSchema = TableSchema.fromBean(Person.class);
```

빌더에서 추상 스키마 사용

다른 접근 방식은 다음 코드에 표시된 대로 각 중첩 bean 클래스에 대해 정적 테이블 스키마 빌더를 사용하는 것입니다.

Address 및 PhoneNumber 클래스의 테이블 스키마는 DynamoDB 테이블과 함께 사용할 수 없다는 점에서 추상적입니다. 기본 키에 대한 정의가 없기 때문입니다. 하지만 Person 클래스의 테이블 스키마에서는 중첩 스키마로 사용됩니다.

PERSON_TABLE_SCHEMA의 정의에서 주석 줄 1과 2줄 뒤에 추상 테이블 스키마를 사용하는 코드가 표시됩니다. EnhanceType.documentOf(...)메서드의 documentOf를 사용한다고 해서 해당 메서드가 확장 문서 API EnhancedDocument 유형을 반환한다는 의미는 아닙니다. 이 컨텍스트의 documentOf(...) 메서드는 테이블 스키마 인수를 사용하여 클래스 인수를 DynamoDB 테이블 속성에 매핑하거나 DynamoDB 테이블 속성에서 클래스 인수를 매핑하는 방법을 알고 있는 객체를 반환합니다.

정적 스키마 코드

```
    // 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.documentOf(Address.class, TABLE_SCHEMA_ADDRESS),
a -> a.name("mainAddress")
                .getter(Person::getMainAddress)
                .setter(Person::setMainAddress))
            .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();
```

복합 유형의 프로젝트 속성

query() 및 scan() 메서드의 경우, addNestedAttributeToProject() 및 attributesToProject()와 같은 메서드 호출을 사용하여 결과에 반환하려는 속성을 지정할 수 있습니다. DynamoDB 향상된 클라이언트 API는 요청을 보내기 전에 Java 메서드 호출 파라미터를 프로젝션 표현식으로 변환합니다.

다음 예제는 Person 테이블을 두 항목으로 채운 다음 세 번의 스캔 작업을 수행합니다.

첫 번째 스캔에서는 결과를 다른 스캔 작업과 비교하기 위해 테이블의 모든 항목에 액세스합니다.

두 번째 스캔에서는 addNestedAttributeToProject() 빌더 메서드를 사용하여 street 속성 값만 반환합니다.

세 번째 스캔 작업에서는 attributesToProject() 빌더 메서드를 사용하여 첫 번째 수준 속성 hobbies에 대한 데이터를 반환합니다. hobbies의 속성 유형은 리스트입니다. 개별 목록 항목에 접근하려면 목록에서 get() 작업을 수행하세요.

```
        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> hobbiesScanResult = personDynamoDbTable.scan(b -> b
                // Use the 'attributesToProject()' method to access first-level
 attributes.
                .attributesToProject("hobbies"));

        hobbiesScanResult.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

`attributesToProject()` 메서드가 프로젝션하려는 속성을 추가하는 다른 빌더 메서드를 따르는 경우 `attributesToProject()`에 제공된 속성 이름 목록이 다른 모든 속성 이름을 대체합니다.

다음 코드 조각의 `ScanEnhancedRequest` 인스턴스를 사용하여 스캔을 수행하면 취미 데이터만 반환됩니다.

```
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]}
```

다음 코드 조각은 `attributesToProject()` 메서드를 먼저 사용합니다. 이 순서는 요청된 다른 모든 속성을 보존합니다.

```
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]}
```

표현식에서 복잡한 유형 사용

참조 해제 연산자를 사용하여 복합 유형의 구조를 탐색해 필터 표현식 및 조건 표현식과 같은 복합 유형을 표현식에 사용할 수 있습니다. 객체 및 맵의 경우 `.` (dot) 및 `[n]`을 사용하는 목록 요소를 사용

합니다(요소의 시퀀스 번호 주변으로 대괄호). 집합의 개별 요소를 참조할 수는 없지만 contains 함수를 사용할 수 있습니다.

다음 예제에서는 스캔 작업에 사용되는 2개의 필터 표현식을 보여줍니다. 필터 표현식은 결과에서 원하는 항목의 일치 조건을 지정합니다. 다음 예제에서는 이전에 표시된 Person, Address, PhoneNumber 클래스를 사용합니다.

```
   public void scanUsingFilterOfNestedAttr() {
       // The following is a filter expression for an attribute that is a map of
Address objects.
       // By using this filter expression, the SDK returns Person objects that have an
address
       // with 'mailing' as a key and 'MS2' for a state value.
       Expression addressFilter = Expression.builder()
               .expression("addresses.#type.#field = :value")
               .putExpressionName("#type", "mailing")
               .putExpressionName("#field", "state")
               .putExpressionValue(":value",
AttributeValue.builder().s("MS2").build())
               .build();

       PageIterable<Person> addressFilterResults = personDynamoDbTable.scan(rb -> rb.
               filterExpression(addressFilter));
       addressFilterResults.items().stream().forEach(p -> logger.info("Person: {}",
p));

       assert addressFilterResults.items().stream().count() == 1;


       // The following is a filter expression for an attribute that is a list of
phone numbers.
       // By using this filter expression, the SDK returns Person objects whose second
phone number
       // in the list has a type equal to 'cell'.
       Expression phoneFilter = Expression.builder()
               .expression("phoneNumbers[1].#type = :type")
               .putExpressionName("#type", "type")
               .putExpressionValue(":type",
AttributeValue.builder().s("cell").build())
               .build();

       PageIterable<Person> phoneFilterResults = personDynamoDbTable.scan(rb -> rb
               .filterExpression(phoneFilter)
```

```
                .attributesToProject("id", "firstName", "lastName", "phoneNumbers")
        );

        phoneFilterResults.items().stream().forEach(p -> logger.info("Person: {}", p));

        assert phoneFilterResults.items().stream().count() == 1;
        assert
phoneFilterResults.items().stream().findFirst().get().getPhoneNumbers().get(1).getType().equal
    }
```

테이블을 채우는 헬퍼 메서드

```
    public static void populateDatabase() {
        Person person1 = new Person();
        person1.setId(1);
        person1.setFirstName("FirstName1");
        person1.setLastName("LastName1");

        Address billingAddr1 = new Address();
        billingAddr1.setState("BS1");
        billingAddr1.setCity("BillingTown1");

        Address mailing1 = new Address();
        mailing1.setState("MS1");
        mailing1.setCity("MailingTown1");

        person1.setAddresses(Map.of("billing", billingAddr1, "mailing", mailing1));

        PhoneNumber pn1_1 = new PhoneNumber();
        pn1_1.setType("work");
        pn1_1.setNumber("111-111-1111");

        PhoneNumber pn1_2 = new PhoneNumber();
        pn1_2.setType("home");
        pn1_2.setNumber("222-222-2222");

        List<PhoneNumber> phoneNumbers1 = List.of(pn1_1, pn1_2);
        person1.setPhoneNumbers(phoneNumbers1);

        personDynamoDbTable.putItem(person1);

        Person person2 = person1;
        person2.setId(2);
```

```
        person2.setFirstName("FirstName2");
        person2.setLastName("LastName2");

        Address billingAddress2 = billingAddr1;
        billingAddress2.setCity("BillingTown2");
        billingAddress2.setState("BS2");

        Address mailing2 = mailing1;
        mailing2.setCity("MailingTown2");
        mailing2.setState("MS2");

        person2.setAddresses(Map.of("billing", billingAddress2, "mailing", mailing2));

        PhoneNumber pn2_1 = new PhoneNumber();
        pn2_1.setType("work");
        pn2_1.setNumber("333-333-3333");

        PhoneNumber pn2_2 = new PhoneNumber();
        pn2_2.setType("cell");
        pn2_2.setNumber("444-444-4444");

        List<PhoneNumber> phoneNumbers2 = List.of(pn2_1, pn2_2);
        person2.setPhoneNumbers(phoneNumbers2);

        personDynamoDbTable.putItem(person2);
    }
```

데이터베이스 내 항목의 JSON 표현

```
{
 "id": 1,
 "addresses": {
  "billing": {
   "city": "BillingTown1",
   "state": "BS1",
   "street": null,
   "zipCode": null
  },
  "mailing": {
   "city": "MailingTown1",
   "state": "MS1",
   "street": null,
   "zipCode": null
```

```
  }
 },
 "firstName": "FirstName1",
 "lastName": "LastName1",
 "phoneNumbers": [
  {
   "number": "111-111-1111",
   "type": "work"
  },
  {
   "number": "222-222-2222",
   "type": "home"
  }
 ]
}

{
 "id": 2,
 "addresses": {
  "billing": {
   "city": "BillingTown2",
   "state": "BS2",
   "street": null,
   "zipCode": null
  },
  "mailing": {
   "city": "MailingTown2",
   "state": "MS2",
   "street": null,
   "zipCode": null
  }
 },
 "firstName": "FirstName2",
 "lastName": "LastName2",
 "phoneNumbers": [
  {
   "number": "333-333-3333",
   "type": "work"
  },
  {
   "number": "444-444-4444",
   "type": "cell"
  }
 ]
```

```
}
```

복잡한 유형이 포함된 항목 업데이트

복잡한 유형이 포함된 항목을 업데이트할 수 있는 2가지 기본 접근 방식이 있습니다.

- 접근 방식 1: 먼저 항목을 검색하고(getItem 사용) 객체를 업데이트한 다음 DynamoDbTable#updateItem을 호출합니다.
- 접근 방식 2: 항목을 검색하지 말고 새 인스턴스를 구문화하고, 업데이트할 속성을 설정하고, 적절한 값의 IgnoreNullsMode를 설정하여 인스턴스를 DynamoDbTable#updateItem에 제출합니다. 이 접근 방식에서는 항목을 업데이트하기 전에 항목을 가져오지 않아도 됩니다.

이 섹션에 표시된 예제에서는 이전에 표시된 Person, Address, PhoneNumber 클래스를 사용합니다.

업데이트 접근 방식 1: 검색 후 업데이트

이 접근 방식을 사용하면 업데이트 시 데이터가 손실되지 않도록 할 수 있습니다. DynamoDB 향상된 클라이언트 API는 복합 유형의 값을 포함하여 DynamoDB에 저장된 항목의 속성을 사용하여 bean을 다시 만듭니다. 그런 다음 getter와 setter 메서드를 사용하여 bean을 업데이트해야 합니다. 이 접근 방식의 단점은 먼저 항목을 검색하는 데 비용이 든다는 것입니다.

다음 예제는 업데이트하기 전에 항목을 처음 검색하는 경우 데이터가 손실되지 않음을 보여줍니다.

```
    public void retrieveThenUpdateExample()  {
        // Assume that we ran this code yesterday.
        Person person = new Person();
        person.setId(1);
        person.setFirstName("FirstName");
        person.setLastName("LastName");

        Address mainAddress = new Address();
        mainAddress.setStreet("123 MyStreet");
        mainAddress.setCity("MyCity");
        mainAddress.setState("MyState");
        mainAddress.setZipCode("MyZipCode");
        person.setMainAddress(mainAddress);

        PhoneNumber homePhone = new PhoneNumber();
        homePhone.setNumber("1111111");
        homePhone.setType("HOME");
```

```
        person.setPhoneNumbers(List.of(homePhone));

        personDynamoDbTable.putItem(person);

        // Assume that we are running this code now.
        // First, retrieve the item
        Person retrievedPerson =
 personDynamoDbTable.getItem(Key.builder().partitionValue(1).build());

        // Make any updates.
        retrievedPerson.getMainAddress().setCity("YourCity");

        // Save the updated bean. 'updateItem' returns the bean as it appears after the
 update.
        Person updatedPerson = personDynamoDbTable.updateItem(retrievedPerson);

        // Verify for this example.
        Address updatedMainAddress = updatedPerson.getMainAddress();
        assert updatedMainAddress.getCity().equals("YourCity");
        assert updatedMainAddress.getState().equals("MyState"); // Unchanged.
        // The list of phone numbers remains; it was not set to null;
        assert updatedPerson.getPhoneNumbers().size() == 1;
    }
```

업데이트 접근 방식 2: 먼저 항목을 검색하지 않고 **`IgnoreNullsMode`** 열거형 사용

DynamoDB에서 항목을 업데이트하려면 업데이트하려는 속성만 있는 새 객체를 제공하고 다른 값은 null로 둘 수 있습니다. 이 접근 방식을 사용하려면 SDK에서 객체의 null 값을 처리하는 방법과 동작을 제어하는 방법을 알고 있어야 합니다.

SDK에서 무시할 null 값 속성을 지정하려면 `UpdateItemEnhancedRequest`를 구축할 때 `IgnoreNullsMode` 열거형을 제공합니다. 열거 값 중 하나를 사용하는 예제로 다음 코드 조각은 `IgnoreNullsMode.SCALAR_ONLY` 모드를 사용합니다.

```
// Create a new Person object to update the existing item in DynamoDB.
Person personForUpdate = new Person();
personForUpdate.setId(1);
personForUpdate.setFirstName("updatedFirstName");  // 'firstName' is a top scalar
 property.

Address addressForUpdate = new Address();
addressForUpdate.setCity("updatedCity");
personForUpdate.setMainAddress(addressForUpdate);
```

```
personDynamoDbTable.updateItem(r -> r
                .item(personForUpdate)
                .ignoreNullsMode(IgnoreNullsMode.SCALAR_ONLY));

/* With IgnoreNullsMode.SCALAR_ONLY provided, The SDK ignores all null properties. The
 SDK adds or replaces
the 'firstName' property with the provided value, "updatedFirstName". The SDK updates
 the 'city' value of
'mainAddress', as long as the 'mainAddress' attribute already exists in DynamoDB.

In the background, the SDK generates an update expression that it sends in the request
 to DynamoDB.
The following JSON object is a simplified version of what it sends. Notice that the SDK
 includes the paths
to 'mainAddress.city' and 'firstName' in the SET clause of the update expression. No
 null values in
'personForUpdate' are included.

{
  "TableName": "PersonTable",
  "Key": {
    "id": {
      "N": "1"
    }
  },
  "ReturnValues": "ALL_NEW",
  "UpdateExpression": "SET #mainAddress.#city = :mainAddress_city, #firstName
 = :firstName",
  "ExpressionAttributeNames": {
    "#city": "city",
    "#firstName": "firstName",
    "#mainAddress": "mainAddress"
  },
  "ExpressionAttributeValues": {
    ":firstName": {
      "S": "updatedFirstName"
    },
    ":mainAddress_city": {
      "S": "updatedCity"
    }
  }
}
```

```
Had we chosen 'IgnoreNullsMode.DEFAULT' instead of 'IgnoreNullsMode.SCALAR_ONLY', the
 SDK would have included
null values in the "ExpressionAttributeValues" section of the request as shown in the
 following snippet.

  "ExpressionAttributeValues": {
    ":mainAddress": {
      "M": {
        "zipCode": {
          "NULL": true
        },
        "city": {
          "S": "updatedCity"
        },
        "street": {
          "NULL": true
        },
        "state": {
          "NULL": true
        }
      }
    },
    ":firstName": {
      "S": "updatedFirstName"
    }
  }
*/
```

업데이트 표현식에 대한 자세한 내용은 Amazon DynamoDB 개발자 안내서를 참조하세요.

`IgnoreNullsMode` 옵션에 대한 설명

- `IgnoreNullsMode.SCALAR_ONLY` - 이 설정을 사용하여 모든 수준에서 스칼라 속성을 업데이트합니다. SDK는 null이 아닌 스칼라 속성만 DynamoDB로 보내는 업데이트 문을 구문화합니다. SDK는 bean 또는 맵의 null 값 스칼라 속성을 무시하여 DynamoDB에 저장된 값을 유지합니다.

  맵 또는 bean의 스칼라 속성을 업데이트할 때 맵이 DynamoDB에 이미 있어야 합니다. DynamoDB의 객체에서 아직 존재하지 않는 객체에 대한 맵 또는 bean을 추가하면 The document path provided in the update expression is invalid for update라는 메시지가 `DynamoDbException`에 표시됩니다. 속성을 업데이트하기 전에 `MAPS_ONLY` 모드를 사용하여 DynamoDB에 bean 또는 맵을 추가해야 합니다.

- `IgnoreNullsMode.MAPS_ONLY` - 이 설정을 사용하여 bean 또는 맵인 속성을 추가하거나 대체합니다. SDK는 객체에 제공된 맵 또는 bean을 대체하거나 추가합니다. 객체에 null인 모든 bean 또는 맵은 무시되어 DynamoDB에 있는 맵이 유지됩니다.
- `IgnoreNullsMode.DEFAULT` - 이 설정을 사용하면 SDK가 null 값을 무시하지 않습니다. null인 모든 수준의 스칼라 속성은 null로 업데이트됩니다. SDK는 객체의 null 값 bean, 맵, 목록 또는 설정 속성을 DynamoDB의 null로 업데이트합니다. 이 모드를 사용하거나 기본 모드이므로 모드를 제공하지 않는 경우, 값을 null로 설정하려는 것이 아니라면 DynamoDB의 값이 업데이트를 위해 객체에 제공된 null로 설정되지 않도록 먼저 항목을 검색해야 합니다.

모든 모드에서 null이 아닌 목록 또는 세트가 있는 `updateItem`에 객체를 제공하면 목록 또는 세트가 DynamoDB에 저장됩니다.

모드가 필요한 이유

객체에 bean을 제공하거나 `updateItem` 메서드에 매핑하면 SDK는 bean의 속성 값(또는 맵의 항목 값)을 사용하여 항목을 업데이트해야 하는지 또는 전체 bean/맵이 DynamoDB에 저장된 항목을 대체해야 하는지 알 수 없습니다.

항목 검색을 먼저 보여주는 이전 예제에서 검색 없이 `mainAddress`의 `city` 속성을 업데이트해 보겠습니다.

```
/* The retrieval example saved the Person object with a 'mainAddress' property whose
 'city' property value is "MyCity".
/* Note that we create a new Person with only the necessary information to update the
 city value
of the mainAddress. */
Person personForUpdate = new Person();
personForUpdate.setId(1);
// The update we want to make changes the city.
Address mainAddressForUpdate = new Address();
mainAddressForUpdate.setCity("YourCity");
personForUpdate.setMainAddress(mainAddressForUpdate);

// Lets' try the following:
Person updatedPerson = personDynamoDbTable.updateItem(personForUpdate);
/*
 Since we haven't retrieved the item, we don't know if the 'mainAddress' property
 already exists, so what update expression should the SDK generate?

A) Should it replace or add the 'mainAddress' with the provided object (setting all
 attributes to null other than city)
```

```
   as shown in the following simplified JSON?

      {
        "TableName": "PersonTable",
        "Key": {
          "id": {
            "N": "1"
          }
        },
        "ReturnValues": "ALL_NEW",
        "UpdateExpression": "SET #mainAddress = :mainAddress",
        "ExpressionAttributeNames": {
          "#mainAddress": "mainAddress"
        },
        "ExpressionAttributeValues": {
          ":mainAddress": {
            "M": {
              "zipCode": {
                "NULL": true
              },
              "city": {
                "S": "YourCity"
              },
              "street": {
                "NULL": true
              },
              "state": {
                "NULL": true
              }
            }
          }
        }
      }

B) Or should it update only the 'city' attribute of an existing 'mainAddress' as shown
 in the following simplified JSON?

      {
        "TableName": "PersonTable",
        "Key": {
          "id": {
            "N": "1"
          }
        },
```

```
        "ReturnValues": "ALL_NEW",
        "UpdateExpression": "SET #mainAddress.#city = :mainAddress_city",
        "ExpressionAttributeNames": {
          "#city": "city",
          "#mainAddress": "mainAddress"
        },
        "ExpressionAttributeValues": {
          ":mainAddress_city": {
            "S": "YourCity"
          }
        }
      }

However, assume that we don't know if the 'mainAddress' already exists. If it doesn't
 exist, the SDK would try to update
an attribute of a non-existent map, which results in an exception.

In this particular case, we would likely select option B (SCALAR_ONLY) to retain the
 other values of the 'mainAddress'.
*/
```

다음 두 예제에서는 MAPS_ONLY 및 SCALAR_ONLY 열거형 값의 사용을 보여줍니다. MAPS_ONLY는 맵을 추가하고 SCALAR_ONLY는 맵을 업데이트합니다.

IgnoreNullsMode.MAPS_ONLY 예제

```
    public void mapsOnlyModeExample() {
        // Assume that we ran this code yesterday.
        Person person = new Person();
        person.setId(1);
        person.setFirstName("FirstName");

        personDynamoDbTable.putItem(person);

        // Assume that we are running this code now.

        /* Note that we create a new Person with only the necessary information to
 update the city value
        of the mainAddress. */
        Person personForUpdate = new Person();
        personForUpdate.setId(1);
        // The update we want to make changes the city.
        Address mainAddressForUpdate = new Address();
```

```
        mainAddressForUpdate.setCity("YourCity");
        personForUpdate.setMainAddress(mainAddressForUpdate);


        Person updatedPerson = personDynamoDbTable.updateItem(r -> r
                .item(personForUpdate)
                .ignoreNullsMode(IgnoreNullsMode.MAPS_ONLY)); // Since the mainAddress
property does not exist, use MAPS_ONLY mode.
        assert updatedPerson.getMainAddress().getCity().equals("YourCity");
        assert updatedPerson.getMainAddress().getState() == null;
    }
```

IgnoreNullsMode.SCALAR_ONLY example

```
    public void scalarOnlyExample() {
        // Assume that we ran this code yesterday.
        Person person = new Person();
        person.setId(1);
        Address mainAddress = new Address();
        mainAddress.setCity("MyCity");
        mainAddress.setState("MyState");
        person.setMainAddress(mainAddress);

        personDynamoDbTable.putItem(person);

        // Assume that we are running this code now.

        /* Note that we create a new Person with only the necessary information to
update the city value
        of the mainAddress. */
        Person personForUpdate = new Person();
        personForUpdate.setId(1);
        // The update we want to make changes the city.
        Address mainAddressForUpdate = new Address();
        mainAddressForUpdate.setCity("YourCity");
        personForUpdate.setMainAddress(mainAddressForUpdate);

        Person updatedPerson = personDynamoDbTable.updateItem(r -> r
                .item(personForUpdate)
                .ignoreNullsMode(IgnoreNullsMode.SCALAR_ONLY)); // SCALAR_ONLY mode
ignores null properties in the in mainAddress.
        assert updatedPerson.getMainAddress().getCity().equals("YourCity");
```

```
        assert updatedPerson.getMainAddress().getState().equals("MyState"); // The
 state property remains the same.
    }
```

각 모드에 대해 무시되는 null 값을 확인하려면 다음 표를 참조하세요. bean 또는 맵으로 작업하는 경우를 제외하고 SCALAR_ONLY 및 MAPS_ONLY로 작업할 수 있습니다.

SDK가 각 모드를 무시하는 **updateItem**에 제출된 객체의 null 값 속성은 무엇인가요?

속성 유형	SCALAR_ONLY 모드	MAPS_ONLY 모드	DEFAULT 모드
상위 스칼라	예	예	아니요
bean 또는 맵	예	예	아니요
bean 또는 맵 항목의 스칼라 값	예[1]	아니요[2]	아니요
목록 또는 세트	예	예	아니요

[1]이는 맵이 DynamoDB에 이미 있다고 가정합니다. 업데이트를 위해 객체에 제공하는 bean 또는 맵의 null이거나 null이 아닌 모든 스칼라 값은 값의 경로가 DynamoDB에 있어야 합니다. SDK는 요청을 제출하기 전에 . (dot) 역참조 연산자를 사용하여 속성에 대한 경로를 구문화합니다.

[2]MAPS_ONLY 모드를 사용하여 bean 또는 맵을 완전히 대체하거나 추가하면 bean 또는 맵의 모든 null 값이 DynamoDB에 저장된 맵에 유지됩니다.

@DynamoDbPreserveEmptyObject을 사용하여 객체를 보존

빈 객체가 있는 Bean을 Amazon DynamoDB에 저장하고 검색 시 SDK가 빈 객체를 다시 생성하도록 하려면 내부 Bean의 접근자에 @DynamoDbPreserveEmptyObject로 주석을 답니다.

주석이 작동하는 방식을 설명하기 위해 코드 예제에서는 다음 두 개의 Bean을 사용합니다.

예시 beans

다음 데이터 클래스에는 두 개의 InnerBean 필드가 있습니다. 접근자 메서드 getInnerBeanWithoutAnno()에는 @DynamoDbPreserveEmptyObject로 주석이 달리지 않습니다. getInnerBeanWithAnno() 메서드에는 주석이 달려 있습니다.

```
@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();
    }
}
```

다음 InnerBean 클래스의 인스턴스는 MyBean의 필드이며 다음 예제 코드에서는 빈 객체로 초기화 됩니다.

```
@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 + '\'' +
                '}';
    }
}
```

다음 코드 예제는 초기화된 내부 bean이 있는 MyBean 객체를 DynamoDB에 저장한 후에 항목을 가져옵니다. 기록된 출력은 innerBeanWithoutAnno는 초기화되지 않았지만 innerBeanWithAnno가 생성되었음을 보여줍니다.

```
   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;
    }
```

대체 정적 스키마

Bean의 주석 대신 다음 StaticTableSchema 버전의 테이블 스키마를 사용할 수 있습니다.

```
    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();
    }
```

중첩된 개체의 null 속성을 저장하지 마세요

@DynamoDbIgnoreNulls 주석을 적용하여 DynamoDB에 데이터 클래스 객체를 저장할 때 중첩된 객체의 null 속성을 건너뛸 수 있습니다. 반대로 null 값이 있는 최상위 속성은 데이터베이스에 저장되지 않습니다.

주석이 작동하는 방식을 설명하기 위해 코드 예제에서는 다음 두 개의 Bean을 사용합니다.

예시 beans

다음 데이터 클래스에는 두 개의 InnerBean 필드가 있습니다. 접근자 메서드 getInnerBeanWithoutAnno()에는 주석이 달려 있지 않습니다. getInnerBeanWithIgnoreNullsAnno() 메서드에는 @DynamoDbIgnoreNulls로 주석이 달려 있습니다.

```
@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();
    }
}
```

다음 InnerBean 클래스의 인스턴스는 MyBean의 필드이며 다음 예제 코드에서 사용됩니다.

```
@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();
    }
}
```

다음 코드 예제에서는 InnerBean 객체를 만들고 객체의 두 속성 중 하나만 값으로 설정합니다.

```
    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);
    }
```

DynamoDB로 전송되는 하위 수준 데이터를 시각화하기 위해 코드는 MyBean 객체를 저장하기 전에 속성 맵을 기록합니다.

로깅된 출력은 innerBeanWithIgnoreNullsAnno가 속성 하나를 출력함을 보여줍니다.

```
innerBeanWithIgnoreNullsAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200)})
```

innerBeanWithoutAnno 인스턴스는 두 개의 속성을 출력합니다. 한 속성의 값은 200이고 다른 하나는 null 값 속성입니다.

```
innerBeanWithoutAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200),
 innerBeanFieldString=AttributeValue(NUL=true)})
```

속성 맵의 JSON 표현

다음 JSON 표현을 사용하면 DynamoDB에 저장된 데이터를 더 쉽게 확인할 수 있습니다.

```
{
  "id": {
    "S": "1"
  },
  "innerBeanWithIgnoreNullsAnno": {
    "M": {
      "innerBeanFieldInteger": {
        "N": "200"
      }
    }
  },
  "innerBeanWithoutAnno": {
    "M": {
      "innerBeanFieldInteger": {
```

```
        "N": "200"
      },
      "innerBeanFieldString": {
        "NULL": true
      }
    }
  }
}
```

대체 정적 스키마

다음 StaticTableSchema 버전의 테이블 스키마를 데이터 클래스 주석에 사용할 수 있습니다.

```
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();
}
```

DynamoDB용 향상된 문서 API를 사용하여 JSON 문서로 작업

AWS SDK for Java 2.x의 향상된 문서 API는 고정된 스키마가 없는 문서 지향 데이터를 처리하도록 설계되었습니다. 하지만 사용자 지정 클래스를 사용하여 개별 속성을 매핑할 수도 있습니다.

향상된 문서 API는 AWS SDK for Java v1.x의 문서 API의 후속 버전입니다.

목차

향상된 문서 API 사용 시작

향상된 문서 API에는 DynamoDB 향상된 클라이언트 API에 필요한 것과 동일한 종속성이 필요합니다. 또한 이 항목의 시작 부분에 표시된 것처럼 `DynamoDbEnhancedClient` 인스턴스도 필요합니다.

확장 문서 API는 AWS SDK for Java 2.x의 버전 2.20.3과 함께 출시되었으므로 해당 버전 이상이 필요합니다.

`DocumentTableSchema` 및 **`DynamoDbTable`** 만들기

향상된 문서 API를 사용하여 DynamoDB 테이블에 대해 명령을 호출하려면 테이블을 클라이언트 측 DynamoDbTable<EnhancedDocument> 리소스 객체와 연결하세요.

향상된 클라이언트의 `table()` 메서드는 `DynamoDbTable<EnhancedDocument>` 인스턴스를 생성하고 DynamoDB 테이블 이름 및 `DocumentTableSchema`에 대한 파라미터를 필요로 합니다.

DocumentTableSchema의 빌더에는 기본 인덱스 키와 하나 이상의 속성 변환기 제공자가 필요합니다. `AttributeConverterProvider.defaultProvider()` 메서드는 기본 유형에 대한 변환기를 제공합니다. 사용자 지정 특성 변환기 공급자를 제공하는 경우에도 지정해야 합니다. 선택적인 보조 인덱스 키를 빌더에 추가할 수 있습니다.

다음 코드 조각은 스키마가 없는 `EnhancedDocument` 객체를 저장하는 `person` DynamoDB 테이블의 클라이언트 측 표현을 생성하는 코드를 보여줍니다.

```
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.
```

다음은 이 단원 전체에서 사용되는 `person` 객체의 JSON 표현을 보여줍니다.

JSON **`person`** 객체

```
{
  "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"
    }
  ]
}
```

향상된 문서 빌드

EnhancedDocument는 속성이 중첩된 복잡한 구조를 가진 문서 유형 객체를 나타냅니다. EnhancedDocument에는 DocumentTableSchema에 지정된 기본 키 속성과 일치하는 최상위 속성이 필요합니다. 나머지 내용은 임의적이며 최상위 속성과 깊이 중첩된 속성으로 구성될 수 있습니다.

요소를 추가하는 여러 가지 방법을 제공하는 빌더를 사용하여 EnhancedDocument 인스턴스를 만듭니다.

JSON 문자열로 빌드

JSON 문자열을 사용하면 호출 한 번으로 EnhancedDocument을 만들 수 있습니다. 다음 코드 조각은 jsonPerson() 헬퍼 메서드에서 반환한 JSON 문자열에서 EnhancedDocument를 생성합니다. jsonPerson() 메서드는 이전에 표시된 사람 객체의 JSON 문자열 버전을 반환합니다.

```
EnhancedDocument document =
        EnhancedDocument.builder()
                        .json( jsonPerson() )
                        .build());
```

개별 요소로 빌드

또는 빌더의 형식이 안전한 메서드를 사용하여 개별 구성 요소에서 EnhancedDocument 인스턴스를 빌드할 수 있습니다.

다음 예제는 이전 예제의 JSON 문자열로 빌드된 향상된 문서와 유사한 person 향상된 문서를 빌드합니다.

```
       /* 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();
```

도우미 메서드

```
    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"));

    }
```

CRUD 작업을 수행

EnhancedDocument 인스턴스를 정의한 후 DynamoDB 테이블에 저장할 수 있습니다. 다음 코드 조각은 개별 요소에서 생성된 PersonDocument를 사용합니다.

```
documentDynamoDbTable.putItem(personDocument);
```

DynamoDB에서 향상된 문서 인스턴스를 읽은 후에는 접근자를 사용하여 개별 속성 값을 추출할 수 있습니다. 다음 코드 조각은 personDocument에서 저장한 데이터에 액세스합니다. 또는 예제 코드의 마지막 부분에 표시된 대로 전체 콘텐츠를 JSON 문자열로 추출할 수 있습니다.

```
        // 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 인스턴스는 매핑된 데이터 클래스 대신 DynamoDbTable의 모든 메서드 또는 DynamoDbEnhancedClient와 함께 사용할 수 있습니다.

향상된 문서 속성을 사용자 지정 객체로 액세스

향상된 문서 API를 사용하면 스키마가 없는 구조의 속성을 읽고 쓸 수 있는 API를 제공할 뿐만 아니라 사용자 정의 클래스의 인스턴스 간에 속성을 변환할 수 있습니다.

향상된 문서 API는 DynamoDB 향상된 클라이언트 API의 일부로 제어 속성 변환 단원에 표시된 AttributeConverterProvider와 AttributeConverter를 사용합니다.

다음 예제는 CustomAttributeConverterProvider를 중첩된 AddressConverter 클래스와 함께 사용하여 Address 객체를 변환합니다.

이 예제는 클래스의 데이터와 필요에 따라 빌드된 구조의 데이터를 혼합할 수 있음을 보여줍니다. 또한 이 예제는 사용자 정의 클래스가 중첩 구조의 모든 수준에서 사용될 수 있음을 보여줍니다. 이 예제의 Address 객체는 맵에서 사용되는 값입니다.

```
   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 코드

```
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 클래스

```
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;
    }
    }
```

주소를 제공하는 도우미 메서드

다음 도우미 메서드는 값에 대한 일반 `Map<String, String>` 인스턴스가 아닌 사용자 지정 `Address` 인스턴스를 값에 사용하는 맵을 제공합니다.

```
    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);
    }
```

DynamoDB를 사용하지 **EnhancedDocument** 않고 사용

일반적으로 EnhancedDocument의 인스턴스를 사용하여 문서 유형 DynamoDB 항목을 읽고 쓰지만 DynamoDB와 독립적으로 사용할 수도 있습니다.

EnhancedDocuments을 사용하면 JSON 문자열이나 사용자 지정 객체를 다음 예제와 같은 AttributeValues의 하위 수준 맵으로 변환하는 기능을 사용할 수 있습니다.

```
    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**
>
> DynamoDB 테이블과 별개로 향상된 문서를 사용하는 경우 빌더에서 속성 변환기 공급자를 명시적으로 설정했는지 확인하세요.
> 반대로 문서 테이블 스키마는 향상된 문서가 DynamoDB 테이블과 함께 사용될 때 변환기 제공자에게 제공합니다.

확장 프로그램을 사용하여 DynamoDB 향상된 클라이언트 작업 사용자 지정

DynamoDB 향상된 클라이언트 API는 매핑 작업 이외의 기능을 제공하는 플러그인 확장을 지원합니다. 확장 프로그램은 2가지 후크 메서드를 사용하여 읽기 및 쓰기 작업 중에 데이터를 수정합니다.

- `beforeWrite()` - 쓰기 작업이 발생하기 전에 수정합니다.
- `afterRead()` - 읽기 작업이 발생한 후 결과를 수정합니다.

일부 작업(예: 항목 업데이트)은 쓰기와 읽기를 차례로 모두 수행하므로 두 후크 메서드가 모두 호출됩니다.

확장 프로그램 로드 방법

확장 프로그램은 향상된 클라이언트 빌더에 지정된 순서대로 로드됩니다. 하나의 확장이 이전 확장에 의해 변환된 값에 대해 작동할 수 있으므로 로드 순서가 중요할 수 있습니다.

기본적으로 향상된 클라이언트는 2개의 확장 프로그램을 로드합니다.

- VersionedRecordExtension - 낙관적 잠금 제공
- AtomicCounterExtension - 카운터 속성을 자동으로 증분

향상된 클라이언트 빌더로 기본 동작을 재정의하고 모든 확장 프로그램을 로드할 수 있습니다. 기본 확장자를 원하지 않는 경우에는 none을 지정할 수도 있습니다.

> **Important**
>
> 자체 확장을 로드하는 경우 확장 클라이언트는 기본 확장을 로드하지 않습니다. 기본 확장 중 하나가 제공하는 동작을 원하는 경우 해당 확장을 확장 목록에 명시적으로 추가해야 합니다.

다음 예제에서는 VersionedRecordExtension 이후 이름이 verifyChecksumExtension인 사용자 지정 확장 프로그램을 로드하는 방법을 보여줍니다. 이 예제에서는 AtomicCounterExtension가 로드되지 않습니다.

```
DynamoDbEnhancedClientExtension versionedRecordExtension =
 VersionedRecordExtension.builder().build();

DynamoDbEnhancedClient enhancedClient =
    DynamoDbEnhancedClient.builder()
                          .dynamoDbClient(dynamoDbClient)
                          .extensions(versionedRecordExtension,
 verifyChecksumExtension)
                          .build();
```

사용 가능한 확장 프로그램 세부 정보 및 구성

다음 섹션에서는 SDK에서 사용 가능한 각 확장 프로그램에 대한 자세한 정보를 제공합니다.

VersionedRecordExtension을 사용한 낙관적 잠금 구현

VersionedRecordExtension 확장 프로그램은 항목이 데이터베이스에 작성될 때 항목 버전 번호를 증분하고 추적하여 낙관적 잠금을 제공합니다. 실제 지속 항목의 버전 번호가 애플리케이션이 마지막으로 읽은 값과 일치하지 않는 경우 쓰기가 실패하도록 하는 조건이 모든 쓰기에 추가됩니다.

구성

항목 버전 번호를 추적하는 데 사용할 속성을 지정하려면 테이블 스키마에서 숫자 속성에 태그를 지정하세요.

다음 코드 조각은 version 속성에 항목 버전 번호가 포함되도록 지정합니다.

```
    @DynamoDbVersionAttribute
    public Integer getVersion() {...};
    public void setVersion(Integer version) {...};
```

동일한 정적 테이블 스키마 접근 방식이 다음 코드 조각에 나와 있습니다.

```
    .addAttribute(Integer.class, a -> a.name("version")
                                       .getter(Customer::getVersion)
                                       .setter(Customer::setVersion)
                                        // Apply the 'version' tag to the attribute.

  .tags(VersionedRecordExtension.AttributeTags.versionAttribute())
```

작동 방법

VersionedRecordExtension이 있는 낙관적 잠금 전략은 이러한 DynamoDbEnhancedClient 및 DynamoDbTable 메서드에 다음과 같은 영향을 끼칩니다.

putItem

새 항목에는 초기 버전 값 0이 할당됩니다. @DynamoDbVersionAttribute(startAt = *X*)로 구성할 수 있습니다.

updateItem

이후 항목을 검색하여 속성을 하나 이상 업데이트한 후 변경 사항을 저장하려고 해도 클라이언트 측과 서버 측의 버전 번호가 일치해야만 작업이 완료됩니다.

작업이 완료되면 버전 번호가 자동으로 1씩 증분됩니다.
@DynamoDbVersionAttribute(incrementBy = *X*)로 구성할 수 있습니다.

deleteItem

DynamoDbVersionAttribute 주석은 영향을 미치지 않습니다. 항목을 삭제할 때 조건 표현식을 수동으로 추가해야 합니다.

다음 예제에서는 조건식을 추가하여 삭제된 항목이 읽은 항목인지 확인합니다. 다음 예제(recordVersion)에서는 @DynamoDbVersionAttribute 주석이 달린 bean의 속성입니다.

```
// 1. Read the item and get its current version.
```

```
Customer item =
 customerTable.getItem(Key.builder().partitionValue("someId").build());
// `recordVersion` is the bean's attribute that is annotated with
 `@DynamoDbVersionAttribute`.
AttributeValue currentVersion = item.getRecordVersion();

// 2. Create conditional delete with the `currentVersion` value.
DeleteItemEnhancedRequest deleteItemRequest =
    DeleteItemEnhancedRequest.builder()
       .key(KEY)
       .conditionExpression(Expression.builder()
           .expression("recordVersion = :current_version_value")
           .putExpressionValue(":current_version_value", currentVersion)
           .build()).build();

customerTable.deleteItem(deleteItemRequest);
```

`transactWriteItems`

- `addPutItem`: 이 메서드는 `putItem`과 동일한 동작을 갖습니다.
- `addUpdateItem`: 이 메서드는 `updateItem`과 동일한 동작을 갖습니다.
- `addDeleteItem`: 이 메서드는 `deleteItem`과 동일한 동작을 갖습니다.

`batchWriteItem`

- `addPutItem`: 이 메서드는 `putItem`과 동일한 동작을 갖습니다.
- `addDeleteItem`: 이 메서드는 `deleteItem`과 동일한 동작을 갖습니다.

> **Note**
>
> DynamoDB 전역 테이블은 동시 업데이트 간에 '최종 라이터 우선 적용' 조정을 사용하며, DynamoDB는 최종 라이터를 확인하기 위해 노력합니다. 전역 테이블을 사용하는 경우 이 '최종 라이터 우선 적용' 정책은 모든 복제본이 결국 DynamoDB에서 결정한 최종 쓰기를 기반으로 수렴되므로 잠금 전략이 예상대로 작동하지 않을 수 있음을 의미합니다.

사용 해제 방법

낙관적 잠금을 사용 해제하려면 `@DynamoDbVersionAttribute` 주석을 사용하지 않습니다.

`AtomicCounterExtension`을 사용하여 카운터 구현

`AtomicCounterExtension` 확장 프로그램은 레코드가 데이터베이스에 기록될 때마다 태그가 지정된 숫자 속성을 증분시킵니다. 시작 및 증분 값을 지정할 수 있습니다. 값을 지정하지 않으면 시작 값은 0으로 설정되고 속성 값은 1씩 증가합니다.

구성

어떤 속성이 카운터인지 지정하려면 테이블 스키마에서 `Long` 유형의 속성에 태그를 지정하세요.

다음 코드 조각은 `counter` 속성의 기본 시작 및 증분 값 사용을 보여줍니다.

```
    @DynamoDbAtomicCounter
    public Long getCounter() {...};
    public void setCounter(Long counter) {...};
```

정적 테이블 스키마 접근 방식이 다음 코드 조각에 나와 있습니다. 원자 카운터 확장은 시작 값 10을 사용하고 레코드가 기록될 때마다 값을 5씩 증가시킵니다.

```
    .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`을 사용하여 타임스탬프 추가

`AutoGeneratedTimestampRecordExtension` 확장 프로그램은 항목이 데이터베이스에 성공적으로 기록될 때마다 `Instant` 유형의 태그가 지정된 속성을 현재 타임스탬프로 자동 업데이트합니다. 이 확장은 기본적으로 로드되지 않습니다.

구성

현재 타임스탬프로 업데이트할 속성을 지정하려면 테이블 스키마에서 `Instant` 속성에 태그를 지정하세요.

`lastUpdate` 속성은 다음 코드 조각에서 확장 프로그램의 동작 대상입니다. 속성이 `Instant` 유형이어야 한다는 요구 사항에 유의하세요.

```
    @DynamoDbAutoGeneratedTimestampAttribute
```

```
    public Instant getLastUpdate() {...}
    public void setLastUpdate(Instant lastUpdate) {...}
```

동일한 정적 테이블 스키마 접근 방식이 다음 코드 조각에 나와 있습니다.

```
     .addAttribute(Instant.class, a -> a.name("lastUpdate")
                                        .getter(Customer::getLastUpdate)
                                        .setter(Customer::setLastUpdate)
                                        // Applying the 'autoGeneratedTimestamp' tag to
the attribute.

.tags(AutoGeneratedTimestampRecordExtension.AttributeTags.autoGeneratedTimestampAttribute())
```

AutoGeneratedUuidExtension을 사용하여 UUID 생성

`AutoGeneratedUuidExtension` 확장 프로그램은 새 레코드가 데이터베이스에 기록될 때 속성에 대한 고유한 UUID(범용 고유 식별자)를 생성합니다. Java JDK UUID.randomUUID() 메서드를 사용하고 `java.lang.String` 유형의 속성에 적용됩니다. 이 확장은 기본적으로 로드되지 않습니다.

구성

`uniqueId` 속성은 다음 코드 조각에서 확장 프로그램의 동작 대상입니다.

```
    @AutoGeneratedUuidExtension
    public String getUniqueId() {...}
    public void setUniqueId(String uniqueId) {...}
```

동일한 정적 테이블 스키마 접근 방식이 다음 코드 조각에 나와 있습니다.

```
     .addAttribute(String.class, a -> a.name("uniqueId")
                                       .getter(Customer::getUniqueId)
                                       .setter(Customer::setUniqueId)
                                       // Applying the 'autoGeneratedUuid' tag to the
attribute.

.tags(AutoGeneratedUuidExtension.AttributeTags.autoGeneratedUuidAttribute())
```

확장 프로그램이 `putItem` 메서드에 한해 UUID를 채우고 `updateItem` 메서드에는 채우지 않도록 하려면 다음 코드 조각과 같이 업데이트 동작 주석을 추가합니다.

```
    @AutoGeneratedUuidExtension
```

```
    @DynamoDbUpdateBehavior(UpdateBehavior.WRITE_IF_NOT_EXISTS)
    public String getUniqueId() {...}
    public void setUniqueId(String uniqueId) {...}
```

정적 테이블 스키마 접근 방식을 사용하는 경우 다음과 같은 동등한 코드를 사용합니다.

```
     .addAttribute(String.class, a -> a.name("uniqueId")
                                       .getter(Customer::getUniqueId)
                                       .setter(Customer::setUniqueId)
                                       // Applying the 'autoGeneratedUuid' tag to the
 attribute.

 .tags(AutoGeneratedUuidExtension.AttributeTags.autoGeneratedUuidAttribute(),

 StaticAttributeTags.updateBehavior(UpdateBehavior.WRITE_IF_NOT_EXISTS))
```

사용자 지정 확장 프로그램 예제

DynamoDbEnhancedClientExtension 인터페이스를 구현하여 사용자 지정 확장 프로그램을 만들 수 있습니다. 다음 사용자 지정 확장 클래스는 데이터베이스의 항목에 아직 속성이 없는 경우 업데이트 표현식을 사용하여 registrationDate 속성을 설정하는 beforeWrite() 메서드를 보여줍니다.

```
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();
    }
}
```

비동기식으로 DynamoDB 향상된 클라이언트 API 사용

애플리케이션에 DynamoDB에 대한 비차단 비동기 호출이 필요한 경우 DynamoDbEnhancedAsyncClient를 사용할 수 있습니다. 동기 구현과 유사하지만 다음과 같은 주요 차이점이 있습니다.

1. DynamoDbEnhancedAsyncClient를 빌드할 때는 다음 코드 조각과 같이 표준 클라이언트의 비동기 버전 DynamoDbAsyncClient을 제공해야 합니다.

```
DynamoDbEnhancedAsyncClient enhancedClient =
    DynamoDbEnhancedAsyncClient.builder()
                               .dynamoDbClient(dynamoDbAsyncClient)
                               .build();
```

2. 단일 데이터 객체를 반환하는 메서드는 결과만 반환하는 대신 결과의 CompletableFuture를 반환합니다. 그러면 애플리케이션은 결과를 차단하지 않고도 다른 작업을 수행할 수 있습니다. 다음 코드 조각은 비동기 getItem() 메서드를 보여줍니다.

```
CompletableFuture<Customer> result = customerDynamoDbTable.getItem(customer);
// Perform other work here.
return result.join();   // Now block and wait for the result.
```

3. 페이지별로 구분된 결과 목록을 반환하는 메서드는 동일한 메서드에 대해 동기식 DynamoDbEnhanceClient를 반환하는 SdkIterable 대신 SdkPublisher를 반환합니다. 그런

다음 애플리케이션은 해당 게시자에 대한 핸들러를 구독하여 차단할 필요 없이 결과를 비동기적으로 처리할 수 있습니다.

```
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.
```

`SdkPublisher` API를 사용한 보다 완전한 예제를 보려면 이 가이드의 비동기 `scan()`메서드를 설명하는 단원의 예제를 참조하세요.

데이터 클래스 주석

다음 표에는 데이터 클래스에 사용할 수 있는 주석이 나열되어 있으며 이 가이드의 정보 및 예제에 대한 링크가 제공됩니다. 테이블은 주석 이름을 기준으로 알파벳 오름차순으로 정렬됩니다.

이 가이드에 사용된 데이터 클래스 주석

주석 이름	주석은 [1]에 적용됩니다.	하는 일	이 가이드에 표시된 위치
DynamoDbAtomicCounter	속성[2]	레코드가 데이터베이스에 기록될 때마다 태그가 지정된 숫자 속성이 증가합니다.	소개 및 토론.
DynamoDbAttribute	속성	DynamoDB 테이블 속성에 매핑되는 Bean 속성을 정의하거나 이름을 바꿉니다.	• 초기 논의. • 시작하기 단원 - 참고 참조. • 쿼리 메서드 예제에서 MovieActor 클래스 내.
DynamoDbAutoGeneratedTimestampAttribute	속성	항목이 데이터베이스에 성공적으로 기록될 때마다 현재 타임스탬프로 태그가 지정된 속성을 업데이트합니다.	소개 및 토론.

주석 이름	주석은 [1]에 적용됩니다.	하는 일	이 가이드에 표시된 위치
DynamoDbAutoGeneratedUuid	속성	새 레코드가 데이터베이스에 기록될 때 속성에 대한 고유한 UUID(범용 고유 식별자)를 생성합니다.	소개 및 토론.
DynamoDbBean	class	데이터 클래스를 테이블 스키마에 매핑할 수 있는 것으로 표시합니다.	먼저 시작하기 단원의 Customer 클래스에서 사용하세요. 가이드 곳곳에 여러 가지 사용법이 나와 있습니다.
DynamoDbConvertedBy	속성	사용자 지정 `AttributeConverter` 을 주석이 달린 속성과 연결합니다.	초기 논의 및 예제.
DynamoDbFlatten	속성	개별 DynamoDB 데이터 클래스의 모든 속성을 평면화하여 데이터베이스에서 읽고 쓰는 레코드에 최상위 속성으로 추가합니다.	• 초기 논의. • 다른 코드에 미치는 영향.
DynamoDbIgnore	속성	그 결과 속성이 매핑되지 않은 상태로 남습니다.	• 초기 논의. • ProductCatalog 클래스에서 사용합니다.
DynamoDbIgnoreNulls	속성	중첩된 DynamoDB 객체의 null 속성이 저장되지 않도록 합니다.	설명 및 예제.

주석 이름	주석은 [1]에 적용됩니다.	하는 일	이 가이드에 표시된 위치
DynamoDbImmutable	class	변경할 수 없는 데이터 클래스를 테이블 스키마에 매핑 가능한 것으로 표시합니다.	• 주석 소개. • ProductCatalog 클래스에서 사용합니다. • Lombok과 함께 사용하세요.
DynamoDbPartitionKey	속성	속성을 DynamoDb 테이블의 기본 파티션 키(해시 키)로 표시합니다.	• 먼저 시작하기 단원의 Customer 클래스 첫 사용 • Lombok과 함께.
DynamoDbPreserveEmptyObject	속성	주석이 달린 속성에 매핑된 개체에 대한 데이터가 없는 경우 개체가 모든 null 필드로 초기화되어야 함을 지정합니다.	설명 및 예제.
DynamoDbSecondaryPartitionKey	속성	속성을 글로벌 보조 인덱스의 파티션 키로 표시합니다.	• 보조 인덱스 및 예제에 사용. • 쿼리 메서드 예제에서. • Lombok 예제에서. • 변경할 수 없는 클래스 사용.

주석 이름	주석은 [1]에 적용됩니다.	하는 일	이 가이드에 표시된 위치
DynamoDbSecondarySortKey	속성	속성을 글로벌 또는 로컬 보조 인덱스의 선택적 정렬 키로 표시합니다.	• 보조 인덱스 및 예제에 사용. • 쿼리 메서드 예제에서. • Lombok 예제에서. • 변경할 수 없는 클래스 사용.
DynamoDbSortKey	속성	속성을 선택적 기본 정렬 키(범위 키)로 표시합니다.	• 고객 클래스의 시작하기 단원. • 변경할 수 없는 클래스 사용. • Lombok 예제에서. • 쿼리 메서드 예제에서.
DynamoDbUpdateBehavior	속성	UpdateItem과 같은 '업데이트' 작업의 일부로 이 속성이 업데이트될 때의 동작을 지정합니다.	소개 및 예제.
DynamoDbVersionAttribute	속성	항목 버전 번호를 증가시킵니다.	소개 및 토론.

[1]속성 수준 주석을 getter 또는 setter에 적용할 수 있지만 둘 다 적용할 수는 없습니다. 이 가이드에서는 getter에 대한 주석을 보여줍니다.

[2]`property` 용어는 일반적으로 JavaBean 데이터 클래스에서 캡슐화된 값에 사용됩니다. 하지만 이 가이드에서는 DynamoDB에서 사용하는 용어와의 일관성을 위해 용어 `attribute`를 대신 사용합니다.

Amazon EC2 작업

이 단원에서는 AWS SDK for Java 2.x를 사용한 Amazon EC2 프로그래밍의 예제를 제공합니다.

주제

- Amazon EC2 인스턴스 관리
- AWS 리전 및 가용 영역 사용
- Amazon EC2의 보안 그룹 작업
- Amazon EC2 인스턴스 메타데이터 작업

Amazon EC2 인스턴스 관리

인스턴스 생성

Ec2Client의 `runInstances` 메서드를 호출하여 사용할 Amazon Machine Image(AMI)와 Amazon EC2 인스턴스 유형을 포함하는 RunInstancesRequest를 제공하여 새 인스턴스를 생성합니다. https://docs.aws.amazon.com//AWSEC2/latest/UserGuide/instance-types.html

가져오기

```
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;
```

코드

```
   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 "";
    }
```

GitHub의 전체 예제를 참조하세요.

인스턴스 시작

Amazon EC2 인스턴스를 시작하려면 Ec2Client의 `startInstances` 메서드를 호출하여 시작할 인스턴스의 ID가 포함된 StartInstancesRequest를 제공합니다.

가져오기

```
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;
```

코드

```
    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);
    }
```

GitHub의 전체 예제를 참조하세요.

인스턴스 중지

Amazon EC2 인스턴스를 중지하려면 Ec2Client의 `stopInstances` 메서드를 호출하여 중지할 인스턴스의 ID가 포함된 StopInstancesRequest를 제공합니다.

가져오기

```
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;
```

코드

```
    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);
    }
```

GitHub의 전체 예제를 참조하세요.

인스턴스 재부팅

Amazon EC2 인스턴스를 재부팅하려면 Ec2Client의 `rebootInstances` 메서드를 호출하여 재부팅할 인스턴스의 ID가 포함된 RebootInstancesRequest를 제공합니다.

가져오기

```
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;
```

코드

```
    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);
     }
  }
```

GitHub의 전체 예제를 참조하세요.

인스턴스 설명

인스턴스를 나열하려면 DescribeInstancesRequest를 생성하고 Ec2Client의 `describeInstances` 메서드를 호출합니다. 계정 및 리전의 Amazon EC2 인스턴스를 나열하는 데 사용할 수 있는 DescribeInstancesResponse 객체를 반환합니다.

인스턴스는 예약별로 그룹화됩니다. 각 예약은 인스턴스를 시작하는 `startInstances` 호출에 해당합니다. 인스턴스를 나열하려면 먼저 `DescribeInstancesResponse` 클래스의 `reservations`를 호출하고 반환된 각 Reservation 객체에서 `instances`를 호출해야 합니다.

가져오기

```
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;
```

코드

```
    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);
        }
```

```
    }
```

결과가 페이징됩니다. 결과 객체의 `nextToken` 메서드에서 반환된 값을 새 요청 객체의 `nextToken` 메서드에 전달하고 다음 번 `describeInstances` 호출의 새 요청 객체를 사용함으로써 추가 결과를 가져올 수 있습니다.

GitHub의 전체 예제를 참조하세요.

인스턴스 모니터링

CPU 및 네트워크 사용률, 사용 가능한 메모리, 남은 디스크 공간 등 Amazon EC2 인스턴스의 다양한 측면을 모니터링할 수 있습니다. 인스턴스 모니터링에 대한 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 모니터링을 Amazon EC2 참조하세요.

인스턴스 모니터링을 시작하려면 모니터링할 인스턴스의 ID로 MonitorInstancesRequest를 생성하고 Ec2Client의 `monitorInstances` 메서드에 전달합니다.

가져오기

```
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;
```

코드

```
    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);
    }
```

GitHub의 전체 예제를 참조하세요.

인스턴스 모니터링 중지

인스턴스 모니터링을 중지하려면 모니터링을 중지할 인스턴스의 ID로 UnmonitorInstancesRequest를 생성하고 Ec2Client의 `unmonitorInstances` 메서드에 전달합니다.

가져오기

```
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;
```

코드

```
    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);
    }
```

GitHub의 전체 예제를 참조하세요.

추가 정보

- Amazon EC2 API 참조의 RunInstances
- Amazon EC2 API 참조의 DescribeInstances
- Amazon EC2 API 참조의 StartInstances
- Amazon EC2 API 참조의 StopInstances
- Amazon EC2 API 참조의 RebootInstances
- Amazon EC2 API 참조의 MonitorInstances
- Amazon EC2 API 참조의 UnmonitorInstances

AWS 리전 및 가용 영역 사용

리전 설명

계정에 사용할 수 있는 리전을 나열하려면 Ec2Client의 `describeRegions` 메서드를 호출합니다. 이 메서드는 DescribeRegionsResponse를 반환합니다. 반환된 객체의 `regions` 메서드를 호출하여 각 리전을 나타내는 Region 객체의 목록을 가져옵니다.

가져오기

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
import java.util.concurrent.CompletableFuture;
```

코드

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeRegionsAndZones {
    public static void main(String[] args) {
        Ec2AsyncClient ec2AsyncClient = Ec2AsyncClient.builder()
            .region(Region.US_EAST_1)
            .build();

        try {
            CompletableFuture<Void> future =
 describeEC2RegionsAndZonesAsync(ec2AsyncClient);
            future.join(); // Wait for both async operations to complete.
        } catch (RuntimeException rte) {
```

```
            System.err.println("An exception occurred: " + (rte.getCause() != null ?
rte.getCause().getMessage() : rte.getMessage()));
        }
    }

    /**
     * Asynchronously describes the EC2 regions and availability zones.
     *
     * @param ec2AsyncClient the EC2 async client used to make the API calls
     * @return a {@link CompletableFuture} that completes when both the region and
availability zone descriptions are complete
     */
    public static CompletableFuture<Void>
describeEC2RegionsAndZonesAsync(Ec2AsyncClient ec2AsyncClient) {
        // Initiate the asynchronous request to describe regions
        CompletableFuture<DescribeRegionsResponse> regionsResponse =
ec2AsyncClient.describeRegions();

        // Handle the response or exception for regions
        CompletableFuture<DescribeRegionsResponse> regionsFuture =
regionsResponse.whenComplete((regionsResp, ex) -> {
            if (ex != null) {
                // Handle the exception by throwing a RuntimeException
                throw new RuntimeException("Failed to describe EC2 regions.", ex);
            } else if (regionsResp == null || regionsResp.regions().isEmpty()) {
                // Throw an exception if the response is null or the result is empty
                throw new RuntimeException("No EC2 regions found.");
            } else {
                // Process the response if no exception occurred and the result is not
empty
                regionsResp.regions().forEach(region -> {
                    System.out.printf(
                        "Found Region %s with endpoint %s%n",
                        region.regionName(),
                        region.endpoint());
                });
            }
        });

        CompletableFuture<DescribeAvailabilityZonesResponse> zonesResponse =
ec2AsyncClient.describeAvailabilityZones();
        CompletableFuture<DescribeAvailabilityZonesResponse> zonesFuture =
zonesResponse.whenComplete((zonesResp, ex) -> {
            if (ex != null) {
```

```
                throw new RuntimeException("Failed to describe EC2 availability
 zones.", ex);
            } else if (zonesResp == null || zonesResp.availabilityZones().isEmpty()) {
                throw new RuntimeException("No EC2 availability zones found.");
            } else {
                zonesResp.availabilityZones().forEach(zone -> {
                    System.out.printf(
                        "Found Availability Zone %s with status %s in region %s%n",
                        zone.zoneName(),
                        zone.state(),
                        zone.regionName()
                    );
                });
            }
        });

        return CompletableFuture.allOf(regionsFuture, zonesFuture);
    }
}
```

GitHub의 전체 예제를 참조하세요.

가용 영역 설명

계정에 사용할 수 있는 각 가용 영역을 나열하려면 Ec2Client의 `describeAvailabilityZones` 메서드를 호출합니다. 이 메서드는 DescribeAvailabilityZonesResponse를 반환합니다. `availabilityZones` 메서드를 호출하여 각 가용 영역을 나타내는 AvailabilityZone 객체의 목록을 가져옵니다.

가져오기

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
import java.util.concurrent.CompletableFuture;
```

코드

Ec2Client를 생성하세요.

```
        Ec2AsyncClient ec2AsyncClient = Ec2AsyncClient.builder()
            .region(Region.US_EAST_1)
```

```
            .build();
```

그런 다음 describeAvailabilityZones()를 호출하고 결과를 검색합니다.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeRegionsAndZones {
    public static void main(String[] args) {
        Ec2AsyncClient ec2AsyncClient = Ec2AsyncClient.builder()
            .region(Region.US_EAST_1)
            .build();

        try {
            CompletableFuture<Void> future =
 describeEC2RegionsAndZonesAsync(ec2AsyncClient);
            future.join(); // Wait for both async operations to complete.
        } catch (RuntimeException rte) {
            System.err.println("An exception occurred: " + (rte.getCause() != null ?
 rte.getCause().getMessage() : rte.getMessage()));
        }
    }

    /**
     * Asynchronously describes the EC2 regions and availability zones.
     *
     * @param ec2AsyncClient the EC2 async client used to make the API calls
     * @return a {@link CompletableFuture} that completes when both the region and
 availability zone descriptions are complete
     */
    public static CompletableFuture<Void>
 describeEC2RegionsAndZonesAsync(Ec2AsyncClient ec2AsyncClient) {
```

```
        // Initiate the asynchronous request to describe regions
        CompletableFuture<DescribeRegionsResponse> regionsResponse =
ec2AsyncClient.describeRegions();

        // Handle the response or exception for regions
        CompletableFuture<DescribeRegionsResponse> regionsFuture =
regionsResponse.whenComplete((regionsResp, ex) -> {
            if (ex != null) {
                // Handle the exception by throwing a RuntimeException
                throw new RuntimeException("Failed to describe EC2 regions.", ex);
            } else if (regionsResp == null || regionsResp.regions().isEmpty()) {
                // Throw an exception if the response is null or the result is empty
                throw new RuntimeException("No EC2 regions found.");
            } else {
                // Process the response if no exception occurred and the result is not
empty
                regionsResp.regions().forEach(region -> {
                    System.out.printf(
                        "Found Region %s with endpoint %s%n",
                        region.regionName(),
                        region.endpoint());
                });
            }
        });

        CompletableFuture<DescribeAvailabilityZonesResponse> zonesResponse =
ec2AsyncClient.describeAvailabilityZones();
        CompletableFuture<DescribeAvailabilityZonesResponse> zonesFuture =
zonesResponse.whenComplete((zonesResp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to describe EC2 availability
zones.", ex);
            } else if (zonesResp == null || zonesResp.availabilityZones().isEmpty()) {
                throw new RuntimeException("No EC2 availability zones found.");
            } else {
                zonesResp.availabilityZones().forEach(zone -> {
                    System.out.printf(
                        "Found Availability Zone %s with status %s in region %s%n",
                        zone.zoneName(),
                        zone.state(),
                        zone.regionName()
                    );
                });
            }
```

```
        });

        return CompletableFuture.allOf(regionsFuture, zonesFuture);
    }
}
```

GitHub의 전체 예제를 참조하세요.

계정 설명

계정에 대한 EC2 관련 정보를 나열하려면 Ec2Client의 `describeAccountAttributes` 메서드를 호출하세요. 이 메서드는 DescribeAccountAttributesResponse 객체를 반환합니다. 이 객체 `accountAttributes` 메서드를 호출하여 AccountAttribute 객체 목록을 확보합니다. 목록 전체를 반복하여 `AccountAttribute` 객체를 검색할 수 있습니다.

`AccountAttribute` 객체의 `attributeValues` 메서드를 간접적으로 호출하여 계정의 속성 값을 확보할 수 있습니다. 이 메서드는 AccountAttributeValue 객체 목록을 반환합니다. 이 두 번째 목록을 반복하여 속성 값을 표시할 수 있습니다(다음 코드 예제 참조).

가져오기

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.DescribeAccountAttributesResponse;
import java.util.concurrent.CompletableFuture;
```

코드

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.DescribeAccountAttributesResponse;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * 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) {
        Ec2AsyncClient ec2AsyncClient = Ec2AsyncClient.builder()
            .region(Region.US_EAST_1)
            .build();

        try {
            CompletableFuture<DescribeAccountAttributesResponse> future =
describeEC2AccountAsync(ec2AsyncClient);
            future.join();
            System.out.println("EC2 Account attributes described successfully.");
        } catch (RuntimeException rte) {
            System.err.println("An exception occurred: " + (rte.getCause() != null ?
rte.getCause().getMessage() : rte.getMessage()));
        }
    }

    /**
     * Describes the EC2 account attributes asynchronously.
     *
     * @param ec2AsyncClient the EC2 asynchronous client to use for the operation
     * @return a {@link CompletableFuture} containing the {@link
DescribeAccountAttributesResponse} with the account attributes
     */
    public static CompletableFuture<DescribeAccountAttributesResponse>
describeEC2AccountAsync(Ec2AsyncClient ec2AsyncClient) {
        CompletableFuture<DescribeAccountAttributesResponse> response =
ec2AsyncClient.describeAccountAttributes();
        return response.whenComplete((accountResults, ex) -> {
            if (ex != null) {
                // Handle the exception by throwing a RuntimeException.
                throw new RuntimeException("Failed to describe EC2 account
attributes.", ex);
            } else if (accountResults == null ||
accountResults.accountAttributes().isEmpty()) {
                // Throw an exception if the response is null or no account attributes
are found.
                throw new RuntimeException("No account attributes found.");
            } else {
                // Process the response if no exception occurred.
                accountResults.accountAttributes().forEach(attribute -> {
                    System.out.println("\nThe name of the attribute is " +
attribute.attributeName());
                    attribute.attributeValues().forEach(
```

```
                        myValue -> System.out.println("The value of the attribute is "
 + myValue.attributeValue()));
                });
            }
        });
    }
}
```

GitHub의 전체 예제를 참조하세요.

추가 정보

- Linux 인스턴스용 Amazon EC2 사용 설명서의 리전 및 가용 영역
- Amazon EC2 API 참조의 DescribeRegions
- Amazon EC2 API 참조의 DescribeAvailabilityZones

Amazon EC2의 보안 그룹 작업

보안 그룹 생성

보안 그룹을 생성하려면 키 이름이 포함된 CreateSecurityGroupRequest를 사용하여 Ec2Client의 `createSecurityGroup` 메서드를 호출하세요.

가져옵니다.

```
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;
```

코드

```
            CreateSecurityGroupRequest createRequest =
 CreateSecurityGroupRequest.builder()
```

```
                .groupName(groupName)
                .description(groupDesc)
                .vpcId(vpcId)
                .build();

            CreateSecurityGroupResponse resp= ec2.createSecurityGroup(createRequest);
```

GitHub의 전체 예제를 참조하세요.

보안 그룹 구성

보안 그룹은 Amazon EC2 인스턴스에 대한 인바운드(수신) 및 아웃바운드(송신) 트래픽을 모두 제어할 수 있습니다.

보안 그룹에 수신 규칙을 추가하려면 AuthorizeSecurityGroupIngressRequest 객체 내에 보안 그룹의 이름과 보안 그룹에 할당하려는 액세스 규칙(IpPermission)을 제공하여 Ec2Client의 `authorizeSecurityGroupIngress` 메서드를 사용합니다. 다음 예제에서는 IP 권한을 보안 그룹에 추가하는 방법을 보여줍니다.

가져옵니다.

```
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;
```

코드

먼저 Ec2Client를 생성합니다.

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

그런 다음 Ec2Client의 `authorizeSecurityGroupIngress` 메서드를 사용하세요.

```
            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 "";
    }
```

송신 규칙을 보안 그룹에 추가하려면 AuthorizeSecurityGroupEgressRequest의 유사한 데이터를 Ec2Client의 `authorizeSecurityGroupEgress` 메서드에 제공합니다.

GitHub의 전체 예제를 참조하세요.

보안 그룹 설명

보안 그룹을 설명하거나 보안 그룹에 대한 정보를 가져오려면 Ec2Client 의 `describeSecurityGroups` 메서드를 호출합니다. 이 메서드는 DescribeSecurityGroupsResponse를 반환하는데, 이를 사용하여 `securityGroups` 메서드를 호출하여 보안 그룹 목록에 액세스할 수 있습니다. 그러면 SecurityGroup 객체 목록이 반환됩니다.

가져옵니다.

```
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;
```

코드

```
    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);
       }
   }
```

GitHub의 전체 예제를 참조하세요.

보안 그룹 삭제

보안 그룹을 삭제하려면 Ec2Client의 `deleteSecurityGroup` 메서드를 호출하여 삭제할 보안 그룹의 ID가 포함된 DeleteSecurityGroupRequest를 이 메서드에 전달합니다.

가져옵니다.

```
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;
```

코드

```
    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);
        }
     }
```

GitHub의 전체 예제를 참조하세요.

추가 정보

- Amazon EC2 Linux 인스턴스용 사용 설명서의 Amazon EC2 보안 그룹
- Linux 인스턴스용 Amazon EC2 사용 설명서의 Linux 인스턴스용 인바운드 트래픽 승인
- Amazon EC2 API 참조의 CreateSecurityGroup

- Amazon EC2 API 참조의 DescribeSecurityGroups
- Amazon EC2 API 참조의 DeleteSecurityGroup
- Amazon EC2 API 참조의 AuthorizeSecurityGroupIngress

Amazon EC2 인스턴스 메타데이터 작업

Amazon EC2 인스턴스 메타데이터 서비스(메타데이터 클라이언트)용 Java SDK 클라이언트를 사용하면 애플리케이션이 로컬 EC2 인스턴스의 메타데이터에 액세스할 수 있습니다. 메타데이터 클라이언트는 IMDSv2(인스턴스 메타데이터 서비스 v2)의 로컬 인스턴스와 함께 작동하며 세션 지향 요청을 사용합니다.

SDK에서는 두 개의 클라이언트 클래스를 사용할 수 있습니다. 동기식은 `Ec2MetadataClient`는 작업 차단용이고 `Ec2MetadataAsyncClient`는 비동기식 비차단 사용 사례용입니다.

시작하기

메타데이터 클라이언트를 사용하려면 `imds` Maven 아티팩트를 프로젝트에 추가하세요. 또한 클래스 경로에 `SdkHttpClient`(또는 비동기 변형용 `SdkAsyncHttpClient`)에 대한 클래스가 필요합니다.

다음 Maven XML은 메타데이터 클라이언트에 대한 종속성과 함께 동기 UrlConnectionHttpClient를 사용하기 위한 종속성 코드 조각을 보여줍니다.

```
<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>
```

Maven 중앙 리포지토리에서 bom 아티팩트의 최신 버전을 검색하세요.

비동기 HTTP 클라이언트를 사용하려면 `url-connection-client` 아티팩트의 종속성 코드 조각을 바꾸세요. 예를 들어 다음 코드 조각은 NettyNioAsyncHttpClient 구현을 가져옵니다.

```
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>netty-nio-client</artifactId>
    </dependency>
```

메타데이터 클라이언트를 사용

메타데이터 클라이언트 인스턴스화

클래스 경로에 `SdkHttpClient` 인터페이스 구현이 하나만 있는 경우 비동기 `Ec2MetadataClient`를 인스턴스화할 수 있습니다. 그러려면 다음 코드 조각과 같이 정적 `Ec2MetadataClient#create()` 메서드를 호출합니다.

```
Ec2MetadataClient client = Ec2MetadataClient.create(); //
 'Ec2MetadataAsyncClient#create' is the asynchronous version.
```

애플리케이션에 `SdkHttpClient` 또는 `SdkHttpAsyncClient` 인터페이스가 여러 개 구현되어 있는 경우 the section called “구성 가능한 HTTP 클라이언트” 단원에 표시된 대로 메타데이터 클라이언트가 사용할 구현을 지정해야 합니다.

> **Note**
>
> Amazon S3와 같은 대부분의 서비스 클라이언트의 경우 Java용 SDK는 `SdkHttpClient` 또는 `SdkHttpAsyncClient` 인터페이스의 구현을 자동으로 추가합니다. 메타데이터 클라이언트가 동일한 구현을 사용하는 경우 `Ec2MetadataClient#create()`가 작동합니다. 다른 구현이 필요한 경우 메타데이터 클라이언트를 만들 때 이를 지정해야 합니다.

요청 전송

인스턴스 메타데이터를 검색하려면 `EC2MetadataClient` 클래스를 인스턴스화하고 인스턴스 메타데이터 카테고리를 지정하는 경로 파라미터를 사용하여 `get` 메서드를 호출합니다.

다음 예제는 `ami-id` 키와 관련된 값을 콘솔에 출력합니다.

```
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.
```

경로가 유효하지 않은 경우 `get` 메서드에서 예외가 발생합니다.

여러 요청에 동일한 클라이언트 인스턴스를 재사용하되 리소스를 릴리스하는 데 더 이상 필요하지 않을 때는 클라이언트에서 `close`를 호출하세요. close 메서드가 호출된 후에는 클라이언트 인스턴스를 더 이상 사용할 수 없습니다.

응답 파싱

EC2 인스턴스 메타데이터는 다양한 형식으로 출력될 수 있습니다. 일반 텍스트와 JSON이 가장 일반적으로 사용되는 형식입니다. 메타데이터 클라이언트는 이러한 형식을 사용할 수 있는 방법을 제공합니다.

다음 예제에서 볼 수 있듯이 `asString` 메서드를 사용하여 데이터를 Java 문자열로 가져옵니다. `asList` 메서드를 사용하여 여러 줄을 반환하는 일반 텍스트 응답을 분리할 수도 있습니다.

```
Ec2MetadataClient client = Ec2MetadataClient.create();
Ec2MetadataResponse response = client.get("/latest/meta-data/");
String fullResponse = response.asString();
List<String> splits = response.asList();
```

응답이 JSON인 경우 다음 코드 조각과 같이 `Ec2MetadataResponse#asDocument` 메서드를 사용하여 JSON 응답을 Document 인스턴스로 파싱합니다.

```
Document fullResponse = response.asDocument();
```

메타데이터 형식이 JSON이 아닌 경우 예외가 발생합니다. 응답이 성공적으로 파싱되면 문서 API를 사용하여 응답을 더 자세히 검사할 수 있습니다. 인스턴스 메타데이터 카테고리 차트를 참조하여 JSON 형식의 응답을 제공하는 메타데이터 카테고리를 알아보세요.

메타데이터 클라이언트 구성

Retries

재시도 메커니즘을 사용하여 메타데이터 클라이언트를 구성할 수 있습니다. 이렇게 하면 클라이언트가 예상치 못한 이유로 실패한 요청을 자동으로 재시도할 수 있습니다. 기본적으로 클라이언트는 시도 사이에 기하급수적인 백오프 시간을 두고 실패한 요청에 대해 세 번 재시도합니다.

사용 사례에 다른 재시도 메커니즘이 필요한 경우 빌더의 `retryPolicy` 메서드를 사용하여 클라이언트를 사용자 지정할 수 있습니다. 예를 들어 다음 예제는 시도 간 고정 지연 시간이 2초이고 재시도 횟수가 5회로 구성된 동기 클라이언트를 보여줍니다.

```
BackoffStrategy fixedBackoffStrategy =
 FixedDelayBackoffStrategy.create(Duration.ofSeconds(2));
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
                     .retryPolicy(retryPolicyBuilder ->
 retryPolicyBuilder.numRetries(5)

 .backoffStrategy(fixedBackoffStrategy))
                     .build();
```

메타데이터 클라이언트에서 사용할 수 있는 여러 BackoffStrategies가 있습니다.

다음 코드 조각과 같이 재시도 메커니즘을 완전히 비활성화할 수도 있습니다.

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
                    .retryPolicy(Ec2MetadataRetryPolicy.none())
                    .build();
```

`Ec2MetadataRetryPolicy#none()`를 사용하면 기본 재시도 정책이 비활성화되어 메타데이터 클라이언트가 재시도를 시도하지 않습니다.

IP 버전

기본적으로 메타데이터 클라이언트는 `http://169.254.169.254`에서 IPV4 엔드포인트를 사용합니다. IPV6 버전을 사용하도록 클라이언트를 변경하려면 빌더의 `endpointMode` 또는 `endpoint` 메서드를 사용하세요. 빌더에서 두 메서드를 모두 호출하면 예외가 발생합니다.

다음 예제는 두 IPV6 옵션을 모두 보여줍니다.

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
                     .endpointMode(EndpointMode.IPV6)
                     .build();
```

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
                     .endpoint(URI.create("http://[fd00:ec2::254]"))
                     .build();
```

주요 기능

비동기 클라이언트

비차단 버전의 클라이언트를 사용하려면 `Ec2MetadataAsyncClient` 클래스의 인스턴스를 인스턴스화하세요. 다음 예제의 코드는 기본 설정으로 비동기 클라이언트를 만들고 `get` 메서드를 사용하여 `ami-id` 키 값을 검색합니다.

```
Ec2MetadataAsyncClient asyncClient = Ec2MetadataAsyncClient.create();
CompletableFuture<Ec2MetadataResponse> response = asyncClient.get("/latest/meta-data/
ami-id");
```

`get` 메서드에서 반환된 `java.util.concurrent.CompletableFuture`은 응답이 반환될 때 완료됩니다. 다음 예제는 `ami-id` 메타데이터를 콘솔에 출력합니다.

```
response.thenAccept(metadata -> System.out.println(metadata.asString()));
```

구성 가능한 HTTP 클라이언트

각 메타데이터 클라이언트의 빌더에는 사용자 지정 HTTP 클라이언트를 제공하는 데 사용할 수 있는 `httpClient` 방법이 있습니다.

다음 예제는 사용자 지정 `UrlConnectionHttpClient` 인스턴스의 코드를 보여줍니다.

```
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.
```

다음 예제는 비동기 메타데이터 클라이언트가 있는 사용자 지정 `NettyNioAsyncHttpClient` 인스턴스의 코드를 보여줍니다.

```
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.
```

이 가이드의 the section called “HTTP 클라이언트 구성” 항목에서는 Java용 SDK에서 사용할 수 있는 HTTP 클라이언트를 구성하는 방법에 대한 세부 정보를 제공합니다.

토큰 캐싱

메타데이터 클라이언트는 IMDSv2를 사용하므로 모든 요청은 세션과 연결됩니다. 세션은 메타데이터 클라이언트가 관리하는 만료일이 있는 토큰으로 정의됩니다. 모든 메타데이터 요청은 만료될 때까지 토큰을 자동으로 재사용합니다.

기본적으로 토큰은 6시간(21,600초) 동안 지속됩니다. 특정 사용 사례에 고급 구성이 필요한 경우가 아니면 기본 TTL(time to live) 값을 유지하는 것이 좋습니다.

필요한 경우 `tokenTtl` 빌더 메서드를 사용하여 기간을 구성하세요. 예를 들어 다음 코드 조각의 코드는 세션 기간이 5분인 클라이언트를 만듭니다.

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
                     .tokenTtl(Duration.ofMinutes(5))
                     .build();
```

빌더에서 `tokenTtl` 메서드 호출을 생략하면 기본 기간인 21,600이 대신 사용됩니다.

작업 IAM

이 섹션에서는 AWS SDK for Java 2.x를 사용한 프로그래밍 AWS Identity and Access Management (IAM)의 예를 제공합니다.

AWS Identity and Access Management (IAM)를 사용하면 사용자의 AWS 서비스 및 리소스에 대한 액세스를 안전하게 제어할 수 있습니다. IAM를 사용하면 AWS 사용자 및 그룹을 생성 및 관리하고 권한을 사용하여 AWS 리소스에 대한 액세스를 허용 및 거부할 수 있습니다. 에 대한 전체 가이드는 IAM 사용 설명서를 IAM참조하세요.

다음 예제에는 각 기술을 보여주는 데 필요한 코드만 포함되어 있습니다. 전체 예제 코드는 GitHub에 있습니다. 이 위치에서 단일 소스 파일을 다운로드하거나 리포지토리를 로컬로 복사하여 모든 예제를 빌드하고 실행할 수 있습니다.

주제

- IAM 액세스 키 관리
- IAM 사용자 관리
- 를 사용하여 IAM 정책 생성 AWS SDK for Java 2.x
- IAM 정책 작업
- IAM 서버 인증서 작업

IAM 액세스 키 관리

액세스 키 생성

IAM 액세스 키를 만들려면 `CreateAccessKeyRequest` 객체를 사용하여 `IamClient's createAccessKey` 메서드를 호출합니다.

> **Note**
>
> IAM은 글로벌 서비스이므로 `IamClient` 호출이 작동하려면 리전을 AWS_GLOBAL로 설정해야 합니다.

가져옵니다.

```
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;
```

코드

```
    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 "";
    }
```

GitHub의 전체 예제를 참조하세요.

액세스 키 나열

해당 사용자의 액세스 키를 나열하려면 키를 나열할 사용자 이름이 포함된 `ListAccessKeysRequest` 객체를 만들어 `IamClient's listAccessKeys` 메서드에 전달합니다.

> **Note**
>
> 사용자 이름을 `listAccessKeys`에 제공하지 않으면 이 메서드는 요청에 서명한 AWS 계정과 연결된 액세스 키를 나열하려고 합니다.

가져옵니다.

```
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;
```

코드

```
    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);
        }
    }
```

listAccessKeys의 결과가 페이징됩니다(호출당 기본 최대 100개 레코드). 반환된 ListAccessKeysResponse 객체에서 isTruncated를 호출하여 쿼리에서 반환된 결과 수가 사용 가능한 것보다 적은지 확인할 수 있습니다. 그런 후 marker에서 ListAccessKeysResponse를 호출해 새 요청 생성 때 사용합니다. 다음 listAccessKeys 호출에 이 새 요청을 사용합니다.

GitHub의 전체 예제를 참조하세요.

액세스 키의 마지막 사용 시간 가져오기

액세스 키가 마지막으로 사용된 시간을 확인하려면 액세스 키의 ID(GetAccessKeyLastUsedRequest 객체를 사용하여 전달할 수 있음)를 사용하여 IamClient's getAccessKeyLastUsed 메서드를 호출합니다.

그런 다음 반환된 GetAccessKeyLastUsedResponse 객체를 사용하여 키의 마지막 사용 시간을 검색할 수 있습니다.

가져옵니다.

```
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;
```

코드

```
    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");
```

```
    }
```

GitHub의 전체 예제를 참조하세요.

액세스 키 활성화 또는 비활성화

UpdateAccessKeyRequest 객체를 만들고 액세스 키 ID, 사용자 이름(선택 사항) 및 원하는 status를 제공한 후, 요청 객체를 IamClient's updateAccessKey 메서드로 전달하여 액세스 키를 활성화하거나 비활성화할 수 있습니다.

가져옵니다.

```
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;
```

코드

```
      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);
        }
    }
```

GitHub의 전체 예제를 참조하세요.

액세스 키 삭제

액세스 키를 영구적으로 삭제하려면 `IamClient`'s `deleteKey` 메서드를 호출하여 액세스 키의 ID와 사용자 이름이 포함된 `DeleteAccessKeyRequest`를 이 메서드에 제공합니다.

> **Note**
>
> 키는 삭제하고 나면 더 이상 가져오거나 사용할 수 없습니다. 나중에 다시 활성화할 수 있도록 키를 일시적으로 비활성화하려면 대신에 `updateAccessKey` 메서드를 사용합니다.

가져옵니다.

```
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;
```

코드

```
    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);
        }
    }
```

GitHub의 전체 예제를 참조하세요.

추가 정보

- IAM API 참조에서 CreateAccessKey
- IAM API 참조에서 ListAccessKeys
- IAM API 참조에서 GetAccessKeyLastUsed
- IAM API 참조에서 UpdateAccessKey
- IAM API 참조에서 DeleteAccessKey

IAM 사용자 관리

사용자 생성

사용자 이름이 포함된 CreateUserRequest 객체를 사용하여 IamClient의 `createUser` 메서드에 사용자 이름을 제공하여 새 IAM 사용자를 생성합니다.

가져옵니다.

```
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;
```

코드

```
    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 "";
    }
```

GitHub의 전체 예제를 참조하세요.

사용자 나열

계정의 IAM 사용자를 나열하려면 새 ListUsersRequest를 생성하여 IamClient의 `listUsers` 메서드에 전달합니다. 반환된 ListUsersResponse 객체의 `users`를 호출하여 사용자 목록을 검색할 수 있습니다.

`listUsers`에서 반환된 사용자 목록이 페이징됩니다. 응답 객체의 `isTruncated` 메서드를 호출하여 가져올 결과가 더 있는지 확인할 수 있습니다. 그런 다음 `true`를 반환하면 응답 객체의 `marker()` 메서드를 호출합니다. 마커 값을 사용해 새 요청 객체를 생성합니다. 그런 다음 새 요청을 사용해 다시 `listUsers` 메서드를 호출합니다.

가져옵니다.

```
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;
```

코드

```
    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);
        }
    }
```

GitHub의 전체 예제를 참조하세요.

사용자 업데이트

사용자를 업데이트하려면 IamClient 객체의 `updateUser` 메서드를 호출합니다. 이 메서드는 사용자 이름 또는 경로를 변경하는 데 사용할 수 있는 UpdateUserRequest 객체를 사용합니다.

가져옵니다.

```
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;
```

코드

```
    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);
        }
      }
```

GitHub의 전체 예제를 참조하세요.

사용자 삭제

사용자를 삭제하려면 삭제할 사용자 이름으로 설정된 UpdateUserRequest 객체를 사용하여 IAMClient의 `deleteUser` 요청을 호출합니다.

가져옵니다.

```
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;
```

코드

```
    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);
        }
    }
```

GitHub의 전체 예제를 참조하세요.

추가 정보

- IAM 사용 설명서의 IAM 사용자
- IAM 사용 설명서의 IAM 사용자 관리
- IAM API 참조의 CreateUser
- IAM API 참조의 ListUsers
- IAM API 참조의 UpdateUser
- IAM API 참조의 DeleteUser

를 사용하여 IAM 정책 생성 AWS SDK for Java 2.x

IAM 정책 빌더 API는 Java에서 IAM 정책을 빌드하고 AWS Identity and Access Management (IAM)에 업로드하는 데 사용할 수 있는 라이브러리입니다.

JSON 문자열을 수동으로 조합하거나 파일을 읽어 IAM 정책을 구축하는 대신 API는 JSON 문자열을 생성하기 위한 클라이언트 측 객체 지향 접근 방식을 제공합니다. JSON 형식의 기존 IAM 정책을 읽으면 API가 이를 IamPolicy 인스턴스로 변환하여 처리합니다.

IAM 정책 빌더 API는 SDK 버전 2.20.105에서 사용할 수 있으므로 Maven 빌드 파일에서 해당 버전 또는 이후 버전을 사용하세요. SDK의 최신 버전 번호는 Maven 센트럴에 나와 있습니다.

다음 코드 조각은 Maven `pom.xml` 파일의 종속성 블록 예제를 보여줍니다. 이를 통해 프로젝트에서 IAM 정책 빌더 API를 사용할 수 있습니다.

```
<dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>iam-policy-builder</artifactId>
    <version>2.27.21</version>
</dependency>
```

`IamPolicy` 생성

이 단원에서는 IAM Policy Builder API를 사용하여 정책을 구축하는 방법에 대한 몇 가지 예제를 보여줍니다.

다음 각 예제에서 `IamPolicy.Builder`로 시작하여 `addStatement` 메서드를 사용하여 명령문을 하나 이상 추가합니다. 이 패턴에 따라 IamStatement.Builder에는 명령문에 효과, 액션, 리소스 및 조건을 추가하는 메서드가 있습니다.

예제: 시간 기반 정책을 생성

다음 예제는 두 시점 사이에 Amazon DynamoDB`GetItem` 작업을 허용하는 자격 증명 기반 정책을 생성합니다.

```
    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 출력

이전 예제의 마지막 문은 다음 JSON 문자열을 반환합니다.

이 예제에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서를 참조하세요.

JSON

```
{
    "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"
            }
        }
    }
}
```

예제: 여러 조건 지정

이 예제는 특정 DynamoDB 속성에 대한 액세스를 허용하는 아이덴티티 기반 정책을 생성하는 방법을 보여줍니다. 정책에는 두 가지 조건이 있습니다.

```
    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 출력

이전 예제의 마지막 문은 다음 JSON 문자열을 반환합니다.

이 예제에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서를 참조하세요.

JSON

```
{
    "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"
            }
        }
    }
}
```

예제: 주체 지정

다음 예제는 조건에 지정된 주체를 제외한 모든 주체의 버킷 액세스를 거부하는 리소스 기반 정책을 생성하는 방법을 보여줍니다.

```
    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 출력

이전 예제의 마지막 문은 다음 JSON 문자열을 반환합니다.

이 예제에 대한 자세한 내용은 AWS Identity and Access Management 사용 설명서를 참조하세요.

JSON

```
{
  "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"
      }
    }
  }
}
```

예제: 교차 계정 액세스를 허용

다음 예제에서는 업로드된 객체에 대한 전체 소유자 제어를 유지하면서 다른이 버킷에 객체를 업로드 AWS 계정 하도록 허용하는 방법을 보여줍니다.

```
    public String allowCrossAccountAccessExample() {
        IamPolicy policy = IamPolicy.builder()
                .addStatement(b -> b
                        .effect(IamEffect.ALLOW)
                        .addPrincipal(IamPrincipalType.AWS, "111122223333")
                        .addAction("s3:PutObject")
                        .addResource("arn:aws:s3:::amzn-s3-demo-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 출력

이전 예제의 마지막 문은 다음 JSON 문자열을 반환합니다.

이 예제에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서를 참조하세요.

JSON

```
{
  "Version":"2012-10-17",
  "Statement" : {
    "Effect" : "Allow",
    "Principal" : {
      "AWS" : "111122223333"
    },
    "Action" : "s3:PutObject",
    "Resource" : "arn:aws:s3:::amzn-s3-demo-bucket/*",
    "Condition" : {
      "StringEquals" : {
        "s3:x-amz-acl" : "bucket-owner-full-control"
      }
    }
  }
}
```

IAM과 함께 `IamPolicy` 사용

`IamPolicy` 인스턴스를 생성한 후에는 `IamClient`를 사용하여 IAM 서비스를 사용할 수 있습니다.

다음 예제는 IAM ID가 `accountID` 파라미터로 지정된 계정의 DynamoDB 테이블에 항목을 쓸 수 있도록 허용하는 정책을 작성합니다. 그러면 정책이 IAM에 JSON 문자열로 업로드됩니다.

```
   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());
    }
```

이 예제는 이전 예제를 기반으로 구축되었습니다. 코드는 정책을 다운로드하고 명령문을 복사하고 변경하여 이를 새 정책의 기초로 사용합니다. 그러면 새 정책이 업로드됩니다.

```
    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

이전 예제는 다음 코드 조각에 표시된 것처럼 생성된 `IamClient` 인수를 사용합니다.

```
IamClient iam = IamClient.builder().region(Region.AWS_GLOBAL).build();
```

JSON 형식의 정책

예제에서는 다음 JSON 문자열을 반환합니다.

```
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"
  }
}
```

IAM 정책 작업

정책 생성

새 정책을 생성하려면 CreatePolicyRequest의 정책 이름과 JSON 형식으로 된 정책 문서를 IamClient의 `createPolicy` 메서드에 제공합니다.

가져옵니다.

```
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;
```

코드

```
    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 "" ;
    }
```

GitHub의 전체 예제를 참조하세요.

정책 가져오기

기존 정책을 검색하려면 GetPolicyRequest 객체 내에 정책의 ARN을 제공하여 IamClient의 `getPolicy` 메서드를 호출하세요.

가져옵니다.

```
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;
```

코드

```
    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);
        }
    }
```

GitHub의 전체 예제를 참조하세요.

역할 정책 연결

IAMClient의 `attachRolePolicy` 메서드를 호출하고 AttachrolePolicyRequest에 역할 이름 및 정책 ARN을 제공하여 정책을 IAM 역할에 연결할 수 있습니다.

가져옵니다.

```
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;
```

코드

```
   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");
    }
```

GitHub의 전체 예제를 참조하세요.

연결된 역할 정책 나열

IamClient의 `listAttachedRolePolicies` 메서드를 호출하여 역할의 연결된 정책을 나열합니다. 이 메서드는 정책을 나열할 역할 이름을 포함하는 ListAttachedRolePoliciesRequest 객체를 사용합니다.

반환된 ListAttachedRolePoliciesResponse 객체에 `getAttachedPolicies`를 호출하여 연결된 정책 목록을 가져옵니다. `ListAttachedRolePoliciesResponse` 객체의 `isTruncated` 메서드가 `true`를 반환하고, `ListAttachedRolePoliciesResponse` 객체의 `marker` 메서드를 호출하는 경우 결과가 잘릴 수 있습니다. 반환된 마커를 사용하여 새 요청을 생성하고, 이를 사용하여 `listAttachedRolePolicies`를 다시 호출해 다음 검색 배치를 가져옵니다.

가져옵니다.

```
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;
```

코드

```
   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");
    }
```

GitHub의 전체 예제를 참조하세요.

역할 정책 분리

역할에서 정책을 분리하려면 IamClient의 `detachRolePolicy` 메서드를 호출하여 DetachRolePolicyRequest의 역할 이름과 정책 ARN을 이 메서드에 지정합니다.

가져옵니다.

```
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;
```

코드

```
    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);
        }
    }
```

GitHub의 전체 예제를 참조하세요.

추가 정보

- IAM 사용 설명서의 IAM 정책 개요.
- IAM 사용 설명서의 AWS IAM 정책 참조.
- IAM API 참조의 CreatePolicy
- IAM API 참조의 GetPolicy
- IAM API 참조의 AttachRolePolicy
- IAM API 참조의 ListAttachedRolePolicies
- IAM API 참조의 DetachRolePolicy

IAM 서버 인증서 작업

AWS에서 웹 사이트나 애플리케이션에 대한 HTTPS 연결을 활성화하려면 SSL/TLS 서버 인증서가 필요합니다. AWS Certificate Manager에서 제공하거나 외부 공급자에게서 얻은 서버 인증서를 사용할 수 있습니다.

서버 인증서를 프로비저닝 및 관리하고 배포할 때 ACM을 사용하는 것이 좋습니다. ACM을 사용하면 인증서를 요청하여 AWS 리소스에 배포할 수 있고, ACM에서 인증서 갱신을 처리하게 할 수 있습니다. ACM에서 제공하는 인증서는 무료입니다. ACM에 대한 추가 정보는 AWS Certificate Manager 사용 설명서를 참조하세요.

서버 인증서 조회

IamClient의 `getServerCertificate` 메서드를 호출하고 인증서 이름이 포함된 GetServerCertificateRequest를 이 메서드에 전달하여 서버 인증서를 검색할 수 있습니다.

가져옵니다.

```
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;
```

코드

```
    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);
        }
    }
```

GitHub의 전체 예제를 참조하세요.

서버 인증서 나열

서버 인증서를 나열하려면 ListServerCertificatesRequest를 사용하여 IamClient의 `listServerCertificates` 메서드를 호출하세요. 그러면 ListServerCertificatesResponse가 반환됩니다.

반환된 `ListServerCertificateResponse` 객체의 `serverCertificateMetadataList` 메서드를 호출하여 각 인증서에 대한 정보를 가져오는 데 사용할 수 있는 ServerCertificateMetadata 객체의 목록을 가져옵니다.

`ListServerCertificateResponse` 객체의 `isTruncated` 메서드가 `true`을 반환하고, `ListServerCertificatesResponse` 객체의 `marker` 메서드를 호출하고, 마커를 사용하여 새 요청을 생성하는 경우 결과가 잘릴 수도 있습니다. 이 경우 새 요청을 사용하여 `listServerCertificates`를 다시 호출해 다음 결과들을 가져옵니다.

가져옵니다.

```
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;
```

코드

```
    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);
        }
    }
```

GitHub의 전체 예제를 참조하세요.

서버 인증서 업데이트

IamClient의 `updateServerCertificate` 메서드를 호출하여 서버 인증서의 이름이나 경로를 업데이트할 수 있습니다. 이 메서드는 서버 인증서의 현재 이름 및 사용할 새 이름이나 새 경로로 설정된 UpdateServerCertificateRequest 객체를 사용합니다.

가져옵니다.

```
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;
```

코드

```
    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);
        }
    }
```

GitHub의 전체 예제를 참조하세요.

서버 인증서 삭제

서버 인증서를 삭제하려면 인증서 이름이 포함된 DeleteServerCertificateRequest와 함께 IamClient의 `deleteServerCertificate` 메서드를 호출하세요.

가져옵니다.

```
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;
```

코드

```
    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);
        }
    }
```

GitHub의 전체 예제를 참조하세요.

추가 정보

- IAM 사용자 설명서에서 서버 인증서 사용
- IAM API 참조의 GetServerCertificate
- IAM API 참조의 ListServerCertificates
- IAM API 참조의 UpdateServerCertificate
- IAM API 참조의 DeleteServerCertificate

- AWS Certificate Manager 사용 설명서

Kinesis 작업

이 단원에서는 AWS SDK for Java 2.x를 사용한 Amazon Kinesis 프로그래밍의 예제를 제공합니다.

Kinesis에 대한 자세한 내용은 Amazon Kinesis 개발자 안내서를 참조하세요.

다음 예제에는 각 기술을 보여주는 데 필요한 코드만 포함되어 있습니다. 전체 예제 코드는 GitHub에 있습니다. 이 위치에서 단일 소스 파일을 다운로드하거나 리포지토리를 로컬로 복사하여 모든 예제를 빌드하고 실행할 수 있습니다.

주제

- Amazon Kinesis Data Streams 가입

Amazon Kinesis Data Streams 가입

다음 예제에서는 `subscribeToShard` 메서드를 사용하여 Amazon Kinesis 데이터 스트림에서 데이터를 검색하고 처리하는 방법을 보여줍니다. Kinesis Data Streams에서는 이제 향상된 팬아웃 기능과 지연 시간이 짧은 HTTP/2 데이터 검색 API를 갖추어 개발자가 동일한 Kinesis 데이터 스트림에서 여러 개의 지연 시간이 짧은 고성능 애플리케이션을 쉽게 실행할 수 있습니다.

설정

먼저 비동기 Kinesis 클라이언트와 SubscribeToShardRequest 객체를 만듭니다. 이러한 객체는 다음 각 예제에서 사용되어 Kinesis 이벤트를 구독합니다.

가져옵니다.

```
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;
```

코드

```
        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();
```

빌더 인터페이스 사용

`builder` 메서드를 사용하여 SubscribeToShardResponseHandler 생성을 간소화할 수 있습니다.

작성기를 사용하여 전체 인터페이스를 구현하는 대신 메서드 호출을 통해 각 수명 주기 콜백을 설정할 수 있습니다.

코드

```
   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);
   }
```

게시자의 더욱 많은 제어를 위해 `publisherTransformer` 메서드를 사용하여 게시자를 사용자 지정할 수 있습니다.

코드

```
   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);
   }
```

GitHub의 전체 예제를 참조하세요.

사용자 지정 응답 핸들러 사용

구독자와 게시자의 완전한 제어를 위해 SubscribeToShardResponseHandler 인터페이스를 구현합니다.

이 예제에서는 onEventStream 메서드를 구현하고, 이는 게시자에 대한 모든 액세스를 허용합니다. 게시자를 구독자가 출력할 이벤트 레코드로 전환하는 방법을 보여줍니다.

코드

```
   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);
    }
```

GitHub의 전체 예제를 참조하세요.

방문자 인터페이스 사용

Visitor 객체를 사용하여 보고 싶은 특정 이벤트를 구독할 수 있습니다.

코드

```
    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);
    }
```

GitHub의 전체 예제를 참조하세요.

사용자 지정 구독자 사용

사용자 지정 구독자를 구현하여 스트림을 구독할 수도 있습니다.

이 코드 조각은 예제 구독자를 보여줍니다.

코드

```
    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");
        }
    }
```

다음 코드 스니펫과 같이 사용자 지정 구독자를 subscribe 메서드에 전달할 수 있습니다.

코드

```
    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);
    }
```

GitHub의 전체 예제를 참조하세요.

Kinesis 데이터 스트림에 데이터 레코드 쓰기

KinesisClient 객체를 이용하면 putRecords 메서드를 사용해 Kinesis 데이터 스트림에 데이터 레코드를 쓸 수 있습니다. 이 메서드를 성공적으로 호출하려면 PutRecordsRequest 객체를 만들어야 합니다. 데이터 스트림 이름을 streamName 메서드에 전달합니다. 또한 다음 코드 예제와 같이 putRecords 메서드를 사용하여 데이터를 전달해야 합니다.

가져옵니다.

```
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;
```

다음 Java 코드 예제에서는 StockTrade 객체가 Kinesis 데이터 스트림에 쓸 데이터로 사용됩니다. 이 예제를 실행하기 전에 데이터 스트림이 생성되었는지 확인합니다.

코드

```
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);
        }
    }
}
```

GitHub의 전체 예제를 참조하세요.

타사 라이브러리 사용

사용자 지정 구독자를 구현하는 대신 타사 라이브러리를 사용할 수 있습니다. 다음은 RxJava 구현을 사용하는 방법에 대한 예입니다. 그렇지만 반응형 스트림 인터페이스를 구현하는 모든 라이브러리를 사용할 수 있습니다. 해당 라이브러리에 대한 자세한 내용은 GitHub의 RxJava 위키 페이지를 참조하십시오.

라이브러리를 사용하려면 종속성으로 추가합니다. 이 예에서는 Maven을 사용하는 경우에 사용할 POM 조각을 알려줍니다.

POM 항목

```
<dependency>
 <groupId>io.reactivex.rxjava2</groupId>
 <artifactId>rxjava</artifactId>
 <version>2.2.21</version>
</dependency>
```

가져옵니다.

```
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;
```

이 예제에서는 `onEventStream` 수명 주기 메서드에서 RxJava를 사용합니다. 이는 게시자에 대한 모든 액세스를 제공하고, 이를 사용하여 Rx Flowable을 생성할 수 있습니다.

코드

```
        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();
```

또한 다음과 같이 `publisherTransformer` 게시자를 포함하여 `Flowable` 메서드를 사용할 수 있습니다. 다음 예제에 표시된 것과 같이 `Flowable` 게시자를 SdkPublisher로 조정해야 합니다.

코드

```
        SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
            .builder()
            .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
            .publisherTransformer(p ->
SdkPublisher.adapt(Flowable.fromPublisher(p).limit(100)))
            .build();
```

GitHub의 전체 예제를 참조하세요.

추가 정보

- Amazon Kinesis API 참조의 SubscribeToShardEvent
- Amazon Kinesis API 참조의 SubscribeToShard

AWS Lambda 함수를 호출, 나열, 삭제

이 단원에서는 AWS SDK for Java 2.x를 사용하여 Lambda 서비스 클라이언트를 통해 프로그래밍하는 예제를 제공합니다.

주제

- Lambda 함수를 호출합니다.
- Lambda 함수 나열
- Lambda 함수 삭제

Lambda 함수를 호출합니다.

`LambdaClient` 객체를 만들고 객체의 `invoke` 메서드를 호출하여 Lambda 함수를 호출할 수 있습니다. `InvokeRequest` 객체를 만들어 Lambda 함수에 전달할 함수 이름 및 페이로드와 같은 추가 정보를 지정합니다. 함수 이름은 arn:aws:lambda:us-east-1:123456789012:function:HelloFunction과 같이 나타납니다. AWS Management 콘솔에서 함수를 확인해 값을 검색할 수 있습니다.

함수에 페이로드 데이터를 전달하려면 정보가 포함된 `SdkBytes` 객체를 만듭니다. 예를 들어 다음 코드 예제에서는 Lambda 함수에 JSON 데이터가 전달됩니다.

가져옵니다.

```
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;
```

코드

다음 코드 예제는 Lambda 함수를 호출하는 방법을 보여줍니다.

```
    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);
        }
    }
```

GitHub의 전체 예제를 참조하세요.

Lambda 함수 나열

`LambdaClient` 객체를 구축하고 `listFunctions` 메서드를 호출합니다. 이 메서드는 `ListFunctionsResponse` 객체를 반환합니다. `FunctionConfiguration` 객체 목록을 반환하도록 이 객체의 `functions` 메서드를 호출할 수 있습니다. 목록을 반복하여 함수에 대한 정보를 검색할 수 있습니다. 예를 들어 다음 Java 코드 예제는 각 함수 이름을 가져오는 방법을 보여줍니다.

가져옵니다.

```
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;
```

코드

다음 Java 코드 예제는 함수 이름 목록을 검색하는 방법을 보여 줍니다.

```
    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);
        }
    }
```

GitHub의 전체 예제를 참조하세요.

Lambda 함수 삭제

LambdaClient 객체를 구축하고 deleteFunction 메서드를 호출합니다. DeleteFunctionRequest 객체를 만들고 deleteFunction 메서드에 전달합니다. 이 개체에는 삭제할 함수의 이름과 같은 정보가 포함되어 있습니다. 함수 이름은 arn:aws:lambda:us-east-1:123456789012:function:HelloFunction과 같이 나타납니다. AWS Management 콘솔에서 함수를 확인해 값을 검색할 수 있습니다.

가져옵니다.

```
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;
```

코드

다음 Java 코드는 Lambda 함수를 삭제하는 방법을 설명합니다.

```
    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);
        }
    }
```

GitHub의 전체 예제를 참조하세요.

Amazon S3와 작업

이 섹션에서는 AWS SDK for Java 2.x를 사용하여 Amazon S3로 작업하는 경우의 배경 정보를 제공합니다. 이 섹션에서는 이 가이드의 코드 예제 섹션에 제시된 Amazon S3 Java v2 예제를 보완합니다.

AWS SDK for Java 2.x의 S3 클라이언트

AWS SDK for Java 2.x는 다양한 유형의 S3 클라이언트를 제공합니다. 다음 표는 차이점을 보여주며 이를 통해 사용 사례에 가장 적합한 것을 결정할 수 있습니다.

Amazon S3 클라이언트의 다양한 종류

S3 클라이언트	간단한 설명	사용해야 하는 경우	제한 및 단점
AWS CRT 기반 S3 클라이언트 인터페이스: S3AsyncClient Builder: S3CrtAsyncClientBuilder	• Java 기반 S3 비동기식 클라이언트와 동일한 비동기식 API 작업을 제공하지만 향상된 성능이 특징입니다. • `aws-crt` 종속성이 필요합니다. • 자동 병렬 전송(멀티파트)을 지원합니다. the section called “고성능 S3 클라이언트 사용”을(를) 참조하세요.	• 애플리케이션에서 대용량 객체(8MB 초과)를 전송하며, 성능 극대화가 필요합니다. • 콘텐츠 길이를 알 수 없는 객체를 업로드하려고 합니다. • 처리량과 성능을 개선하는 향상된 연결 풀링 및 DNS 로드 밸런싱이 필요합니다. • 네트워크 장애가 발생할 경우 전송 신뢰	• Java 기반 S3 클라이언트보다 적은 구성 설정을 지원합니다. • 추가 종속성이 필요합니다.

S3 클라이언트	간단한 설명	사용해야 하는 경우	제한 및 단점
		성을 개선해야 합니다. 각 실패한 부분은 전송을 처음부터 다시 시작하지 않고 재시도됩니다.	
멀티파트가 사용 설정된 Java 기반 S3 비동기식 클라이언트 인터페이스: S3AsyncClient Builder: S3AsyncClientBuilder	• 비동기식 API를 제공합니다. • 만들 때 멀티파트를 사용 설정하면 자동 병렬 전송(멀티파트)이 지원됩니다. the section called “병렬 전송 지원 구성”을(를) 참조하세요.	• 애플리케이션에서 대용량 객체를 전송하며, 성능 향상이 필요합니다. • 콘텐츠 길이를 알 수 없는 객체를 업로드하려고 합니다. • 네트워크 장애가 발생할 경우 전송 신뢰성을 개선해야 합니다. 각 실패한 부분은 전송을 처음부터 다시 시작하지 않고 재시도됩니다. • AWS CRT 기반 S3 클라이언트에서 사용할 수 없는 구성 옵션이 필요합니다.	AWS CRT 기반 S3 클라이언트보다 성능이 떨어집니다.
멀티파트가 사용 설정되지 않은 Java 기반 S3 비동기식 클라이언트 인터페이스: S3AsyncClient Builder: S3AsyncClientBuilder	• 비동기식 API를 제공합니다.	• 8MB 미만의 객체를 전송하고 있습니다. • 비동기식 API를 원합니다.	성능 최적화가 필요하지 않습니다.

S3 클라이언트	간단한 설명	사용해야 하는 경우	제한 및 단점
Java 기반 S3 동기식 클라이언트 인터페이스: S3Client Builder: S3ClientBuilder	• 동기식 API를 제공합니다.	• 8MB 미만의 객체를 전송하고 있습니다. • 동기식 API가 필요합니다.	성능 최적화가 필요하지 않습니다.

> **Note**
>
> 버전 2.18.x 이후부터는 엔드포인트 재정의를 포함할 때 AWS SDK for Java 2.x에서 가상 호스트식 주소 지정을 사용합니다. 이는 버킷 이름이 유효한 DNS 레이블인 한 적용됩니다. `true`에서 `forcePathStyle` 메서드를 호출하여 클라이언트가 버킷에 경로 스타일 주소 지정을 사용하도록 강제합니다.
> 다음 예제는 엔드포인트 재정의 및 경로 스타일 주소 지정을 사용하여 구성된 서비스 클라이언트를 보여줍니다.
>
> ```
> S3Client client = S3Client.builder()
>                           .region(Region.US_WEST_2)
>                           .endpointOverride(URI.create("https://s3.us-
> west-2.amazonaws.com"))
>                           .forcePathStyle(true)
>                           .build();
> ```

주제

- AWS SDK for Java 2.x를 사용하여 Amazon S3에 스트림 업로드
- 미리 서명된 Amazon S3 URL로 작업
- Amazon S3를 위한 교차 리전 액세스
- 체크섬을 통한 데이터 무결성 보호
- 고성능 S3 클라이언트 사용: AWS CRT 기반 S3 클라이언트
- 병렬 전송을 사용하도록 Java 기반 S3 비동기식 클라이언트 구성
- Amazon S3 Transfer Manager로 파일 및 디렉터리 전송

- S3 이벤트 알림 작업

AWS SDK for Java 2.x를 사용하여 Amazon S3에 스트림 업로드

스트림을 사용하여 putObject 또는 uploadPart를 사용하여 S3에 콘텐츠를 업로드하는 경우 동기식 API의 RequestBody 팩토리 클래스를 사용하여 스트림을 제공합니다. 비동기식 API의 경우 AsyncRequestBody는 동등한 팩토리 클래스입니다.

스트림을 업로드하는 방법

동기식 API의 경우 다음과 같은 RequestBody의 팩토리 메서드를 사용하여 스트림을 제공할 수 있습니다.

- `fromInputStream(InputStream inputStream, long contentLength)`

  `fromContentProvider(ContentStreamProvider provider, long contentLength, String mimeType)`

  - ContentStreamProvider에는 `fromInputStream(InputStream inputStream)` 팩토리 메서드가 있습니다.
- `fromContentProvider(ContentStreamProvider provider, String mimeType)`

비동기식 API의 경우 다음과 같은 AsyncRequestBody의 팩토리 메서드를 사용할 수 있습니다.

- `fromInputStream(InputStream inputStream, Long contentLength, ExecutorService executor)`
- `fromInputStream(AsyncRequestBodyFromInputStreamConfiguration configuration)`
  - AsyncRequestBodyFromInputStreamConfiguration.Builder를 사용하여 스트림을 제공합니다.
- `fromInputStream(Consumer<AsyncRequestBodyFromInputStreamConfiguration.Builder> configuration)`
- `forBlockingInputStream(Long contentLength)`
  - 결과 BlockingInputStreamAsyncRequestBody에는 스트림을 제공하는 데 사용할 수 있는 `writeInputStream(InputStream inputStream)` 메서드가 포함되어 있습니다.

업로드 수행

스트림의 길이를 알고 있는 경우

이전에 표시된 메서드의 서명에서 볼 수 있듯이 대부분의 메서드는 콘텐츠 길이 파라미터를 허용합니다.

바이트 단위로 콘텐츠 길이를 알고 있는 경우 정확한 값을 입력합니다.

```
// Always provide the exact content length when it's available.
long contentLength = 1024; // Exact size in bytes.
s3Client.putObject(req -> req
    .bucket("amzn-s3-demo-bucket")
    .key("my-key"),
RequestBody.fromInputStream(inputStream, contentLength));
```

> **Warning**
>
> 입력 스트림에서 업로드할 때 지정된 콘텐츠 길이가 실제 바이트 수와 일치하지 않으면 다음과 같은 상황이 발생할 수 있습니다.
>
> - 잘린 객체 - 지정된 길이가 너무 작을 경우
> - 업로드 실패 또는 연결 중단 - 지정된 길이가 너무 큰 경우

스트림 길이를 모르는 경우

동기식 API 사용

`fromContentProvider(ContentStreamProvider provider, String mimeType)`를 사용합니다.

```
public PutObjectResponse syncClient_stream_unknown_size(String bucketName, String key,
 InputStream inputStream) {

    S3Client s3Client = S3Client.create();

    RequestBody body =
 RequestBody.fromContentProvider(ContentStreamProvider.fromInputStream(inputStream),
 "text/plain");
    PutObjectResponse putObjectResponse = s3Client.putObject(b ->
 b.bucket(BUCKET_NAME).key(KEY_NAME), body);
```

```
    return putObjectResponse;
}
```

SDK는 메모리의 전체 스트림을 버퍼링하여 콘텐츠 길이를 계산하므로 대용량 스트림에서 메모리 문제가 발생할 수 있습니다. 동기식 클라이언트로 대용량 스트림을 업로드해야 하는 경우 멀티파트 API를 사용하는 것이 좋습니다.

동기식 클라이언트 API 및 멀티파트 API를 사용하여 스트림 업로드

```
public static void uploadStreamToS3(String bucketName, String key, InputStream
 inputStream) {
    // Create S3 client
    S3Client s3Client = S3Client.create();
    try {
        // Step 1: Initiate the multipart upload
        CreateMultipartUploadRequest createMultipartUploadRequest =
 CreateMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

        CreateMultipartUploadResponse createResponse =
 s3Client.createMultipartUpload(createMultipartUploadRequest);
        String uploadId = createResponse.uploadId();
        System.out.println("Started multipart upload with ID: " + uploadId);

        // Step 2: Upload parts
        List<CompletedPart> completedParts = new ArrayList<>();
        int partNumber = 1;
        byte[] buffer = new byte[PART_SIZE];
        int bytesRead;

        try {
            while ((bytesRead = readFullyOrToEnd(inputStream, buffer)) > 0) {
                // Create request to upload a part
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                        .bucket(bucketName)
                        .key(key)
                        .uploadId(uploadId)
                        .partNumber(partNumber)
                        .build();

                // If we didn't read a full buffer, create a properly sized byte array
```

```
                RequestBody requestBody;
                if (bytesRead < PART_SIZE) {
                    byte[] lastPartBuffer = new byte[bytesRead];
                    System.arraycopy(buffer, 0, lastPartBuffer, 0, bytesRead);
                    requestBody = RequestBody.fromBytes(lastPartBuffer);
                } else {
                    requestBody = RequestBody.fromBytes(buffer);
                }

                // Upload the part and save the response's ETag
                UploadPartResponse uploadPartResponse =
s3Client.uploadPart(uploadPartRequest, requestBody);
                CompletedPart part = CompletedPart.builder()
                        .partNumber(partNumber)
                        .eTag(uploadPartResponse.eTag())
                        .build();
                completedParts.add(part);

                System.out.println("Uploaded part " + partNumber + " with size " +
bytesRead + " bytes");
                partNumber++;
            }

            // Step 3: Complete the multipart upload
            CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
                    .parts(completedParts)
                    .build();

            CompleteMultipartUploadRequest completeRequest =
CompleteMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .multipartUpload(completedMultipartUpload)
                    .build();

            CompleteMultipartUploadResponse completeResponse =
s3Client.completeMultipartUpload(completeRequest);
            System.out.println("Multipart upload completed. Object URL: " +
completeResponse.location());

        } catch (Exception e) {
            // If an error occurs, abort the multipart upload
```

```
            System.err.println("Error during multipart upload: " + e.getMessage());
            AbortMultipartUploadRequest abortRequest =
 AbortMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .build();
            s3Client.abortMultipartUpload(abortRequest);
            System.err.println("Multipart upload aborted");
        } finally {
            try {
                inputStream.close();
            } catch (IOException e) {
                System.err.println("Error closing input stream: " + e.getMessage());
            }
        }
    } finally {
        s3Client.close();
    }
}

/**
 * Reads from the input stream into the buffer, attempting to fill the buffer
 completely
 * or until the end of the stream is reached.
 *
 * @param inputStream the input stream to read from
 * @param buffer      the buffer to fill
 * @return the number of bytes read, or -1 if the end of the stream is reached before
 any bytes are read
 * @throws IOException if an I/O error occurs
 */
private static int readFullyOrToEnd(InputStream inputStream, byte[] buffer) throws
 IOException {
    int totalBytesRead = 0;
    int bytesRead;

    while (totalBytesRead < buffer.length) {
        bytesRead = inputStream.read(buffer, totalBytesRead, buffer.length -
 totalBytesRead);
        if (bytesRead == -1) {
            break; // End of stream
        }
        totalBytesRead += bytesRead;
```

```
    }

    return totalBytesRead > 0 ? totalBytesRead : -1;
}
```

> **Note**
>
> 대부분의 사용 사례에서는 크기를 알 수 없는 스트림에 비동기식 클라이언트 API를 사용하는 것이 좋습니다. 이 접근 방식은 병렬 전송을 사용하고 더 간단한 프로그래밍 인터페이스를 제공합니다. 스트림이 큰 경우 SDK가 스트림 분할을 멀티파트 청크로 처리하기 때문입니다. 멀티파트가 사용 설정된 표준 S3 비동기식 클라이언트와 AWS CRT 기반 S3 클라이언트 모두 이 접근 방식을 구현합니다. 다음 섹션에서는 이 접근 방식의 예제를 보여줍니다.

동기식 API 사용

`fromInputStream(InputStream inputStream, Long contentLength, ExecutorService executor)`에 대한 `contentLength` 인수의 `null` 값을 제공할 수 있습니다.

Example AWS CRT 기반 비동기식 클라이언트 사용:

```
public PutObjectResponse crtClient_stream_unknown_size(String bucketName, String key,
 InputStream inputStream) {

    S3AsyncClient s3AsyncClient = S3AsyncClient.crtCreate();
    ExecutorService executor = Executors.newSingleThreadExecutor();
    AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream, null,
 executor);  // 'null' indicates that the

     // content length is unknown.
    CompletableFuture<PutObjectResponse> responseFuture =
            s3AsyncClient.putObject(r -> r.bucket(bucketName).key(key), body)
                    .exceptionally(e -> {
                        if (e != null){
                            logger.error(e.getMessage(), e);
                        }
                        return null;
                    });

    PutObjectResponse response = responseFuture.join(); // Wait for the response.
    executor.shutdown();
    return response;
```

```
}
```

Example멀티파트가 사용 설정된 표준 비동기식 클라이언트 사용:

```
public PutObjectResponse asyncClient_multipart_stream_unknown_size(String bucketName,
 String key, InputStream inputStream) {

    S3AsyncClient s3AsyncClient =
 S3AsyncClient.builder().multipartEnabled(true).build();
    ExecutorService executor = Executors.newSingleThreadExecutor();
    AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream, null,
 executor); // 'null' indicates that the

    // content length is unknown.
    CompletableFuture<PutObjectResponse> responseFuture =
            s3AsyncClient.putObject(r -> r.bucket(bucketName).key(key), body)
                    .exceptionally(e -> {
                        if (e != null) {
                            logger.error(e.getMessage(), e);
                        }
                        return null;
                    });

    PutObjectResponse response = responseFuture.join(); // Wait for the response.
    executor.shutdown();
    return response;
}
```

미리 서명된 Amazon S3 URL로 작업

미리 서명된 URL은 사용자에게 AWS 보안 인증이나 권한이 없어도 비공개 S3 객체에 대한 임시 액세스를 제공합니다.

예를 들어 Alice가 S3 객체에 대한 액세스 권한을 가지고 있고 해당 객체에 대한 액세스 권한을 Bob과 일시적으로 공유하려고 할 경우, Alice는 미리 서명된 GET 요청을 생성하여 Bob과 공유할 수 있으므로 Bob은 Alice의 보안 인증에 액세스하지 않고도 객체를 다운로드할 수 있습니다. HTTP GET 요청과 HTTP PUT 요청에 대해 미리 서명된 URL을 생성할 수 있습니다.

객체에 대해 미리 서명된 URL을 생성한 다음 다운로드(GET 요청)합니다.

다음 예시는 두 부분으로 구성되어 있습니다.

- 1부: Alice가 객체의 미리 서명된 URL을 생성합니다.
- 2부: Bob은 미리 서명된 URL을 사용하여 객체를 다운로드합니다.

1부: URL 생성

Alice는 이미 S3 버킷에 객체를 보유하고 있으며 다음 코드를 사용하여 Bob이 후속 GET 요청에서 사용할 수 있는 URL 문자열을 생성합니다.

가져오기

```
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();
        }
    }
```

2부: 객체 다운로드

Bob은 다음 세 가지 코드 옵션 중 하나를 사용하여 객체를 다운로드합니다. 또는 브라우저를 사용하여 GET 요청을 수행할 수도 있습니다.

JDK **`HttpURLConnection`**(v1.1 이후) 사용

```
    /* 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();
    }
```

JDK **HttpClient**(v11 이후) 사용

```
    /* 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();
    }
```

SDK for Java의 **SdkHttpClient** 사용

```
    /* Use the AWS SDK for Java SdkHttpClient class to do the download. */
    public byte[] useSdkHttpClientToGet(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();
    }
```

GitHub의 전체 예제 및 테스트를 참조하세요.

업로드를 위해 미리 서명된 URL을 생성한 다음 파일을 업로드(PUT 요청)합니다.

다음 예시는 두 부분으로 구성되어 있습니다.

- 1부: Alice는 객체를 업로드하기 위해 미리 서명된 URL을 생성합니다.
- 2부: Bob이 미리 서명된 URL을 사용하여 파일을 업로드합니다.

1부: URL 생성

Alice는 이미 S3 버킷을 보유하고 있으며 다음 코드를 사용하여 Bob이 후속 PUT 요청에서 사용할 수 있는 URL 문자열을 생성합니다.

가져오기

```
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();
       }
   }
```

2부: 파일 객체 업로드

Bob은 다음 세 가지 코드 옵션 중 하나를 사용하여 파일을 업로드합니다.

JDK **`HttpURLConnection`**(v1.1 이후) 사용

```
   /* 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);
        }
    }
```

JDK **HttpClient**(v11 이후) 사용

```
    /* 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);
        }
    }
```

SDK for Java의 **SdkHttpClient** 사용

```
    /* 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);
        }
    }
```

GitHub의 전체 예제 및 테스트를 참조하세요.

Amazon S3를 위한 교차 리전 액세스

Amazon Simple Storage Service(Amazon S3) 버킷으로 작업하면 일반적으로 버킷의 AWS 리전를 알 수 있습니다. 사용하는 리전전은 S3 클라이언트를 생성할 때 결정됩니다.

하지만 특정 버킷으로 작업해야 하는데 해당 버킷이 S3 클라이언트에 설정된 동일한 리전전에 있는지 알 수 없는 경우가 있습니다.

버킷 리전전을 결정하기 위해 더 많은 호출을 하는 대신 SDK를 사용하여 여러 리전전의 S3 버킷에 액세스할 수 있도록 할 수 있습니다.

설정

SDK `2.20.111` 버전에서 교차 리전 액세스에 대한 지원을 사용할 수 있게 되었습니다. 다음 코드 조각과 같이 Maven 빌드 파일의 `s3` 종속 항목에 대해 이 버전 또는 이후 버전을 사용하세요.

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>s3</artifactId>
  <version>2.27.21</version>
</dependency>
```

다음으로 S3 클라이언트를 생성할 때 코드 조각에 표시된 대로 교차 리전 액세스를 활성화하세요. 기본적으로 액세스가 활성화되어 있지 않습니다.

```
S3AsyncClient client = S3AsyncClient.builder()
                                    .crossRegionAccessEnabled(true)
                                    .build();
```

SDK가 교차 리전 액세스를 제공하는 방법

`putObject` 메서드를 사용할 때와 같이 요청에서 기존 버킷을 참조하면 SDK가 클라이언트용으로 구성된 리전전에 대한 요청을 시작합니다.

특정 리전전에 버킷이 없는 경우 오류 응답에는 버킷이 있는 실제 리전전이 포함됩니다. 그러면 SDK는 두 번째 요청에서 올바른 리전전을 사용합니다.

동일한 버킷에 대한 향후 요청을 최적화하기 위해 SDK는 이 리전전 매핑을 클라이언트에 캐시합니다.

고려 사항

교차 리전 버킷 액세스를 활성화하는 경우, 버킷이 클라이언트의 구성된 리전전에 있지 않으면 첫 번째 API 호출 시 지연 시간이 늘어날 수 있다는 점에 유의하세요. 하지만 후속 호출은 캐시된 리전전 정보를 활용하므로 성능이 향상됩니다.

교차 리전 액세스를 활성화해도 버킷 액세스는 영향을 받지 않습니다. 사용자는 버킷이 상주하는 리전전에 상관없이 버킷에 액세스할 수 있는 권한을 부여받아야 합니다.

체크섬을 통한 데이터 무결성 보호

Amazon Simple Storage Service(S3)는 객체를 업로드할 때 체크섬을 지정하는 기능을 제공합니다. 체크섬을 지정하면 객체와 함께 저장되며 객체를 다운로드할 때 유효성을 검사할 수 있습니다.

체크섬은 파일을 전송할 때 데이터 무결성을 한층 더 강화합니다. 체크섬을 사용하면 수신된 파일이 원본 파일과 일치하는지 확인하여 데이터 일관성을 확인할 수 있습니다. Amazon S3의 체크섬에 대한 자세한 내용은 지원되는 알고리즘을 포함한 Amazon Simple Storage Service 사용 설명서를 참조하세요.

필요에 가장 적합한 알고리즘을 유연하게 선택하고 SDK가 체크섬을 계산하도록 할 수 있습니다. 또는 지원되는 알고리즘 중 하나를 사용하여 미리 계산된 체크섬 값을 제공할 수 있습니다.

> **Note**
>
> AWS SDK for Java 2.x의 버전 2.30.0부터 SDK는 업로드에 대한 CRC32 체크섬을 자동으로 계산하여 기본 무결성 보호를 제공합니다. 사전 계산된 체크섬 값을 제공하지 않거나 SDK가 체크섬을 계산하는 데 사용해야 하는 알고리즘을 지정하지 않은 경우 SDK는 이 체크섬을 계산하는 데 사용됩니다.
> 또한 SDK는 AWS SDK 및 도구 참조 안내서에서 확인할 수 있고 외부에서 설정할 수 있는 데이터 무결성 보호에 대한 전역 설정을 지원합니다.

체크섬은 객체 업로드와 객체 다운로드라는 두 가지 요청 단계로 설명합니다.

객체 업로드

`putObject` 메서드를 사용하여 객체를 업로드하고 체크섬 알고리즘을 제공하면 SDK가 지정된 알고리즘의 체크섬을 계산합니다.

다음 코드 조각은 SHA256 체크섬이 있는 객체를 업로드하라는 요청을 보여줍니다. SDK는 요청을 보내면 SHA256 체크섬을 계산하고 객체를 업로드합니다. Amazon S3는 체크섬을 계산하고 SDK에서 제

공하는 체크섬과 비교하여 콘텐츠의 무결성을 확인합니다. Amazon S3는 객체와 함께 체크섬을 저장합니다.

```
public void putObjectWithChecksum() {
        s3Client.putObject(b -> b
                .bucket(bucketName)
                .key(key)
                .checksumAlgorithm(ChecksumAlgorithm.SHA256),
            RequestBody.fromString("This is a test"));
}
```

요청에 체크섬 알고리즘을 제공하지 않는 경우 체크섬 동작은 다음 표와 같이 사용하는 SDK 버전에 따라 달라집니다.

체크섬 알고리즘이 제공되지 않은 경우 체크섬 동작

Java SDK 버전	체크섬 동작
2.30.0 이하	SDK는 CRC 기반 체크섬을 자동으로 계산하여 요청에 제공하지 않습니다.
2.30.0 이상	SDK는 CRC32 알고리즘을 사용하여 체크섬을 계산하고 요청에 제공합니다. Amazon S3는 자체 CRC32 체크섬을 계산하여 전송의 무결성을 확인하고 이를 SDK에서 제공하는 체크섬과 비교합니다. 체크섬이 일치하면 체크섬이 객체와 함께 저장됩니다.

미리 계산된 체크섬 값 사용

요청과 함께 제공되는 사전 계산된 체크섬 값은 SDK의 자동 계산을 비활성화하고 제공된 값을 대신 사용합니다.

다음 예제는 사전 계산된 SHA256 체크섬을 포함하는 요청을 보여줍니다.

```
    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)));
    }
```

Amazon S3에서 체크섬 값이 지정된 알고리즘에 대해 올바르지 않다고 판단하면 서비스는 오류 응답을 반환합니다.

멀티파트 업로드

멀티파트 업로드에 체크섬을 사용할 수도 있습니다.

Java 2.x용 SDK는 멀티파트 업로드에 체크섬을 사용하는 두 가지 옵션을 제공합니다. 첫 번째 옵션은 `S3TransferManager`를 사용합니다.

다음 전송 관리자 예제는 업로드를 위한 SHA1 알고리즘을 지정합니다.

```
    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();
    }
```

업로드에 전송 관리자를 사용할 때 체크섬 알고리즘을 제공하지 않으면 SDK는 CRC32 알고리즘을 기반으로 체크섬을 자동으로 계산합니다. SDK는 SDK의 모든 버전에서 이 계산을 수행합니다.

두 번째 옵션은 S3Client API(또는 S3AsyncClient API)를 사용하여 멀티파트 업로드를 수행합니다. 이 접근법으로 추가 체크섬을 지정하는 경우 업로드를 시작할 때 사용할 알고리즘을 지정해야 합니다. 또한 각 파트 요청에 대한 알고리즘을 지정하고 업로드 후 각 파트에 대해 계산된 체크섬을 제공해야 합니다.

```
    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()));
    }
```

전체 예제 코드 및 테스트 코드는 GitHub 코드 예제 저장소에 있습니다.

객체 다운로드

getObject 메서드를 사용하여 객체를 다운로드하면, `GetObjectRequest`용 빌더의 `checksumMode` 메서드가 `ChecksumMode.ENABLED`로 설정된 경우.

다음 스니펫의 요청은 체크섬을 계산하고 값을 비교하여 응답의 체크섬을 검증하도록 SDK에 지시합니다.

```
    public GetObjectResponse getObjectWithChecksum() {
        return s3Client.getObject(b -> b
                        .bucket(bucketName)
                        .key(key)
                        .checksumMode(ChecksumMode.ENABLED))
                .response();
    }
```

> **Note**
>
> 체크섬과 함께 객체를 업로드하지 않은 경우 검증이 수행되지 않습니다.

기타 체크섬 계산 옵션

> **Note**
>
> 전송된 데이터의 데이터 무결성을 확인하고 전송 오류를 식별하려면 사용자가 체크섬 계산 옵션에 대한 SDK 기본 설정을 유지하는 것이 좋습니다. 기본적으로 SDK는 `PutObject` 및 `GetObject`를 포함한 많은 S3 작업에 대해 이 중요한 검사를 추가합니다.

그러나 Amazon S3를 사용하려면 최소한의 체크섬 확인이 필요한 경우 기본 구성 설정을 변경하여 많은 검사를 비활성화할 수 있습니다.

필요하지 않은 경우 자동 체크섬 계산 비활성화

`PutObject` 및 `GetObject`와 같이 이를 지원하는 작업에 대해 SDK에서 자동 체크섬 계산을 비활성화할 수 있습니다. 그러나 일부 S3 작업에는 체크섬 계산이 필요하므로 이러한 작업에 대한 체크섬 계산을 비활성화할 수 없습니다.

SDK는 요청의 페이로드와 응답의 페이로드에 대한 체크섬 계산을 위한 별도의 설정을 제공합니다.

다음 목록은 다양한 범위에서 체크섬 계산을 최소화하는 데 사용할 수 있는 설정을 설명합니다.

- 모든 애플리케이션 범위 - 환경 변수 또는 공유 AWS `config` 및 `credentials` 파일의 프로파일에서 설정을 변경하면 모든 애플리케이션에서 이러한 설정을 사용할 수 있습니다. 이러한 설정은 애플리케이션 또는 서비스 클라이언트 범위에서 재정의되지 않는 한 모든 AWS SDK 애플리케이션의 모든 서비스 클라이언트에 영향을 줍니다.
  - 프로파일에 설정을 추가합니다.

    ```
    [default]
    request_checksum_calculation = WHEN_REQUIRED
    response_checksum_validation = WHEN_REQUIRED
    ```

  - 환경 변수를 추가합니다.

    ```
    AWS_REQUEST_CHECKSUM_CALCULATION=WHEN_REQUIRED
    AWS_RESPONSE_CHECKSUM_VALIDATION=WHEN_REQUIRED
    ```

- 현재 애플리케이션 범위 - Java 시스템 속성(`aws.requestChecksumCalculation`)을 `WHEN_REQUIRED`로 설정하여 체크섬 계산을 제한할 수 있습니다. 응답에 해당하는 시스템 속성은 `aws.responseChecksumValidation`입니다.

이러한 설정은 서비스 클라이언트를 만드는 중에 재정의되지 않는 한 애플리케이션의 모든 SDK 서비스 클라이언트에 영향을 줍니다.

애플리케이션 시작 시 시스템 속성을 설정합니다.

```
import software.amazon.awssdk.core.SdkSystemSetting;
import software.amazon.awssdk.core.checksums.RequestChecksumCalculation;
import software.amazon.awssdk.core.checksums.ResponseChecksumValidation;
import software.amazon.awssdk.services.s3.S3Client;

class DemoClass {
    public static void main(String[] args) {


 System.setProperty(SdkSystemSetting.AWS_REQUEST_CHECKSUM_CALCULATION.property(), //
 Resolves to "aws.requestChecksumCalculation".
                "WHEN_REQUIRED");

 System.setProperty(SdkSystemSetting.AWS_RESPONSE_CHECKSUM_VALIDATION.property(), //
 Resolves to "aws.responseChecksumValidation".
                "WHEN_REQUIRED");

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

        // Use s3Client.
    }
}
```

- 단일 S3 서비스 클라이언트 범위 - 빌더 메서드를 사용하여 최소 체크섬 양을 계산하도록 단일 S3 서비스 클라이언트를 구성할 수 있습니다.

```
import software.amazon.awssdk.core.checksums.RequestChecksumCalculation;
import software.amazon.awssdk.services.s3.S3Client;

public class RequiredChecksums {
    public static void main(String[] args) {
        S3Client s3 = S3Client.builder()
                .requestChecksumCalculation(RequestChecksumCalculation.WHEN_REQUIRED)
                .responseChecksumValidation(ResponseChecksumValidation.WHEN_REQUIRED)
                .build();

        // Use s3Client.
```

```
    }
// ...
}
```

간소화된 MD5 호환성을 위해 **`LegacyMd5Plugin`** 사용

버전 2.30.0의 CRC32 체크섬 동작 릴리스와 함께 SDK는 필요한 작업에 대한 MD5 체크섬 계산을 중단했습니다.

S3 작업에 레거시 MD5 체크섬 동작이 필요한 경우 SDK 버전 2.31.32에서 릴리스된 `LegacyMd5Plugin`을 사용할 수 있습니다.

`LegacyMd5Plugin`은 레거시 MD5 체크섬 동작에 의존하는 애플리케이션과의 호환성을 유지해야 할 때 유용합니다. 특히 S3A 파일 시스템 커넥터(Apache Spark, Iceberg)와 함께 사용되는 서드 파티 S3 호환 스토리지 공급자와 함께 작업할 때 유용합니다.

`LegacyMd5Plugin`을 사용하려면 S3 클라이언트 빌더에 추가합니다.

```
// For synchronous S3 client.
S3Client s3Client = S3Client.builder()
                            .addPlugin(LegacyMd5Plugin.create())
                            .build();

// For asynchronous S3 client.
S3AsyncClient asyncClient = S3AsyncClient.builder()
                                         .addPlugin(LegacyMd5Plugin.create())
                                         .build();
```

체크섬이 필요한 작업에 MD5 체크섬을 추가하고 체크섬을 지원하지만 필요하지 않은 작업에 대한 SDK 기본 체크섬 추가를 건너뛰려면 `ClientBuilder` 옵션 `requestChecksumCalculation` 및 `responseChecksumValidation`을 `WHEN_REQUIRED`로 사용할 수 있습니다. 이렇게 하면 체크섬이 필요한 작업에만 SDK 기본 체크섬이 추가됩니다.

```
// Use the `LegacyMd5Plugin` with `requestChecksumCalculation` and
 `responseChecksumValidation` set to WHEN_REQUIRED.
S3AsyncClient asyncClient = S3AsyncClient.builder()
                                         .addPlugin(LegacyMd5Plugin.create())

 .requestChecksumCalculation(RequestChecksumCalculation.WHEN_REQUIRED)

 .responseChecksumValidation(ResponseChecksumValidation.WHEN_REQUIRED)
```

```
                .build();
```

이 구성은 최신 체크섬 알고리즘을 완전히 지원하지는 않지만 특정 작업에 대해 여전히 MD5 체크섬이 필요한 서드 파티 S3 호환 스토리지 시스템으로 작업할 때 특히 유용합니다.

고성능 S3 클라이언트 사용: AWS CRT 기반 S3 클라이언트

AWS Common Runtime(CRT)을 기반으로 구축된 AWS CRT 기반 S3 클라이언트는 대체 S3 비동기 클라이언트입니다. Amazon S3의 멀티파트 업로드 API와 바이트 범위 가져오기를 자동으로 사용하여 향상된 성능과 안정성을 바탕으로 Amazon Simple Storage Service(Amazon S3)와 객체를 주고 받습니다.

AWS CRT 기반 S3 클라이언트는 네트워크 장애 발생 시 전송 안정성을 개선합니다. 전송을 처음부터 다시 시작하지 않고 파일 전송의 실패한 개별 부분을 다시 시도하여 안정성이 향상됩니다.

또한 AWS CRT 기반 S3 클라이언트는 향상된 연결 폴링 및 DNS(도메인 이름 시스템) 로드 밸런싱을 제공하여 처리량도 개선합니다.

SDK의 표준 S3 비동기 클라이언트 대신 AWS CRT 기반 S3 클라이언트를 사용하고 개선된 처리량을 즉시 활용할 수 있습니다.

> **Important**
>
> AWS CRT 기반 S3 클라이언트는 현재 클라이언트 수준이나 요청 수준에서 SDK 지표 컬렉션을 지원하지 않습니다.

SDK의 AWS CRT 기반 구성 요소

이 항목에 설명된 AWS CRT 기반 S3 클라이언트와 AWS CRT 기반 HTTP 클라이언트는 SDK의 서로 다른 구성 요소입니다.

AWS CRT 기반 S3 클라이언트는 S3AsyncClient 인터페이스를 구현한 것으로, Amazon S3 서비스를 사용하는 데 사용됩니다. 이는 `S3AsyncClient` 인터페이스의 Java 기반 구현의 대안이며 여러 가지 이점을 제공합니다.

AWS CRT 기반 HTTP 클라이언트는 SdkAsyncHttpClient 인터페이스를 구현한 것으로, 일반 HTTP 통신에 사용됩니다. 이는 `SdkAsyncHttpClient` 인터페이스의 Netty 구현의 대안이며 여러 가지 이점을 제공합니다.

두 구성 요소 모두 AWS 공용 런타임의 라이브러리를 사용하지만 AWS CRT 기반 S3 클라이언트는 aws-c-s3 라이브러리를 사용하고 S3 멀티파트 업로드 API 기능을 지원합니다. AWS CRT 기반 HTTP 클라이언트는 범용이므로 S3 멀티파트 업로드 API 기능을 지원하지 않습니다.

AWS CRT 기반 S3 클라이언트 사용을 위한 종속성 추가

AWS CRT 기반 S3 클라이언트를 사용하려면 Maven 프로젝트 파일에 다음 두 종속성을 추가하세요. 예제는 사용하는 최소 버전을 보여 줍니다. Maven 중앙 리포지토리에서 가장 최신 버전의 s3 및 aws-crt 아티팩트를 검색하세요.

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>s3</artifactId>
  <version>2.27.21</version>
</dependency>
<dependency>
  <groupId>software.amazon.awssdk.crt</groupId>
  <artifactId>aws-crt</artifactId>
  <version>0.30.11</version>
</dependency>
```

AWS CRT-based S3 클라이언트의 인스턴스를 만들기

다음 코드 조각과 같이 기본 설정을 사용하여 AWS CRT 기반 S3 클라이언트의 인스턴스를 생성합니다.

```
S3AsyncClient s3AsyncClient = S3AsyncClient.crtCreate();
```

클라이언트를 구성하려면 AWS CRT 클라이언트 빌더를 사용하세요. 빌더 방법을 변경하여 표준 S3 비동기 클라이언트에서 AWS CRT 기반 클라이언트로 전환할 수 있습니다.

```
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**
>
> 표준 빌더의 일부 설정은 AWS CRT 클라이언트 빌더에서 현재 지원되지 않을 수 있습니다. `S3AsyncClient#builder()`를 호출하여 표준 빌더를 가져오세요.

AWS CRT 기반 S3 클라이언트를 사용

AWS CRT 기반 S3 클라이언트를 사용하여 Amazon S3 API 작업을 호출합니다. 다음 예제는 AWS SDK for Java를 통해 사용할 수 있는 PutObject 및 GetObject 작업을 보여줍니다.

```
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();
```

알 수 없는 크기의 스트림 업로드

AWS AWS CRT 기반 S3 클라이언트의 한 가지 중요한 이점은 크기가 알려지지 않은 입력 스트림을 효율적으로 처리할 수 있다는 것입니다. 이는 총 크기를 미리 확인할 수 없는 소스에서 데이터를 업로드해야 할 때 특히 유용합니다.

```
public PutObjectResponse crtClient_stream_unknown_size(String bucketName, String key,
 InputStream inputStream) {

    S3AsyncClient s3AsyncClient = S3AsyncClient.crtCreate();
    ExecutorService executor = Executors.newSingleThreadExecutor();
    AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream, null,
 executor);  // 'null' indicates that the

     // content length is unknown.
    CompletableFuture<PutObjectResponse> responseFuture =
            s3AsyncClient.putObject(r -> r.bucket(bucketName).key(key), body)
                    .exceptionally(e -> {
                        if (e != null){
                            logger.error(e.getMessage(), e);
                        }
                        return null;
                    });

    PutObjectResponse response = responseFuture.join(); // Wait for the response.
    executor.shutdown();
    return response;
}
```

이 기능은 잘못된 콘텐츠 길이 사양으로 인해 객체가 잘리거나 업로드가 실패할 수 있는 기존 업로드의 일반적인 문제를 방지하는 데 도움이 됩니다.

구성 제한 사항

AWS CRT 기반 S3 클라이언트와 Java 기반 S3 비동기식 클라이언트는 성능 엣지를 제공하는 AWS CRT 기반 S3 클라이언트와 유사한 기능을 제공합니다. 그러나 AWS CRT 기반 S3 클라이언트에는 Java 기반 S3 비동기식 클라이언트에 있는 구성 설정이 없습니다. 이러한 설정은 다음과 같습니다.

- 클라이언트 수준 구성: API 호출 시도 제한 시간, 압축 실행 인터셉터, 지표 게시자, 사용자 지정 실행 속성, 사용자 지정 고급 옵션, 사용자 지정된 예약 실행기 서비스, 사용자 지정 헤더
- 요청 수준 구성: 사용자 지정 서명자, 자격 증명 공급자, API 호출 시도 제한 시간

구성 차이의 전체 목록은 API 참조를 참조하세요.

Java 기반 S3 비동기식 클라이언트	AWS CRT 기반 S3 클라이언트
클라이언트 수준 구성 • ClientOverrideConfiguration.Builder	클라이언트 수준 구성 • S3CrtAsyncClientBuilder
요청 수준 구성 • RequestOverrideConfiguration.Builder • AwsRequestOverrideConfiguration.Builder	요청 수준 구성 없음

병렬 전송을 사용하도록 Java 기반 S3 비동기식 클라이언트 구성

버전 2.27.5부터 표준 Java 기반 S3 비동기식 클라이언트는 자동 병렬 전송(멀티파트 업로드 및 다운로드)을 지원합니다. Java 기반 S3 비동기식 클라이언트를 만들 때 병렬 전송에 대한 지원을 구성합니다.

이 섹션에서는 병렬 전송을 사용하는 방법과 구성을 사용자 지정하는 방법을 보여줍니다.

`S3AsyncClient`의 인스턴스를 만듭니다.

빌더에서 `multipart*` 메서드를 호출하지 않고 `S3AsyncClient` 인스턴스를 만들면 병렬 전송이 사용되지 않습니다. 다음 각 문은 멀티파트 업로드 및 다운로드를 지원하지 않고 Java 기반 S3 비동기식 클라이언트를 만듭니다.

멀티파트 지원 없이 만들기

Example

```
import software.amazon.awssdk.auth.credentials.ProcessCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3AsyncClient;


S3AsyncClient s3Client = S3AsyncClient.create();

S3AsyncClient s3Client2 = S3AsyncClient.builder().build();
```

```
S3AsyncClient s3Client3 = S3AsyncClient.builder()
        .credentialsProvider(ProcessCredentialsProvider.builder().build())
        .region(Region.EU_NORTH_1)
        .build();
```

멀티파트 지원으로 만들기

기본 설정으로 병렬 전송을 사용하려면 다음 예제와 같이 빌더에서 `true`를 호출하고 `multipartEnabled`에 전달합니다.

Example

```
S3AsyncClient s3AsyncClient2 = S3AsyncClient.builder()
        .multipartEnabled(true)
        .build();
```

`thresholdInBytes` 및 `minimumPartSizeInBytes` 설정의 기본값은 8MiB입니다.

멀티파트 설정을 사용자 지정하면 다음과 같이 병렬 전송이 자동으로 사용됩니다.

Example

```
import software.amazon.awssdk.services.s3.S3AsyncClient;
import static software.amazon.awssdk.transfer.s3.SizeConstant.MB;


S3AsyncClient s3AsyncClient2 = S3AsyncClient.builder()
        .multipartConfiguration(b -> b
                .thresholdInBytes(16 * MB)
                .minimumPartSizeInBytes(10 * MB))
        .build();
```

알 수 없는 크기의 스트림 업로드

멀티파트가 사용 설정된 Java 기반 S3 비동기식 클라이언트는 총 크기를 미리 알 수 없는 입력 스트림을 효율적으로 처리할 수 있습니다.

```
public PutObjectResponse asyncClient_multipart_stream_unknown_size(String bucketName,
 String key, InputStream inputStream) {

    S3AsyncClient s3AsyncClient =
 S3AsyncClient.builder().multipartEnabled(true).build();
```

```
    ExecutorService executor = Executors.newSingleThreadExecutor();
    AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream, null,
 executor); // 'null' indicates that the

    // content length is unknown.
    CompletableFuture<PutObjectResponse> responseFuture =
            s3AsyncClient.putObject(r -> r.bucket(bucketName).key(key), body)
                    .exceptionally(e -> {
                        if (e != null) {
                            logger.error(e.getMessage(), e);
                        }
                        return null;
                    });

    PutObjectResponse response = responseFuture.join(); // Wait for the response.
    executor.shutdown();
    return response;
}
```

이 접근 방식은 잘린 객체 또는 실패한 업로드와 같이 잘못된 콘텐츠 길이를 수동으로 지정할 때 발생할 수 있는 문제를 방지합니다.

Amazon S3 Transfer Manager로 파일 및 디렉터리 전송

Amazon S3 Transfer Manager는 AWS SDK for Java 2.x를 위한 오픈 소스의 고급 파일 전송 유틸리티입니다. 이를 사용하여 Amazon Simple Storage Service(Amazon S3)와 파일 및 디렉터리를 주고받을 수 있습니다.

AWS CRT 기반 S3 클라이언트 또는 표준 Java 기반 S3 비동기식 클라이언트(멀티파트 사용 설정)를 기반으로 구축된 S3 Transfer Manager는 멀티파트 업로드 및 바이트 범위 가져오기와 같은 성능 개선의 이점을 활용할 수 있습니다.

S3 Transfer Manager를 사용하면 전송 진행 상황을 실시간으로 모니터링하고 나중에 실행하기 위해 전송을 일시 중지할 수도 있습니다.

시작하기

빌드 파일에 종속성을 추가

향상된 멀티파트 성능으로 S3 Transfer Manager를 사용하려면 필요한 종속성으로 빌드 파일을 구성합니다.

Use the AWS CRT-based S3 client

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>bom</artifactId>
            <version>2.27.21[1]</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>0.29.143[2]</version>
    </dependency>
</dependencies>
```

[1] 최신 버전. [2] 최신 버전.

Use the Java-based S3 async client

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>bom</artifactId>
            <version>2.27.21[1]</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
<dependencies>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>s3-transfer-manager</artifactId>
```

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

[1] 최신 버전.

S3 Transfer Manager 인스턴스를 생성

병렬 전송을 사용하려면 AWS CRT 기반 S3 클라이언트 또는 멀티파트가 사용 설정된 Java 기반 S3 비동기식 클라이언트를 전달해야 합니다. 다음 예제에서는 사용자 지정 설정으로 S3 Transfer Manager를 구성하는 방법을 보여줍니다.

Use the AWS CRT-based S3 client

```
        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();
```

Use the Java-based S3 async client

`aws-crt` 종속성이 빌드 파일에 포함되지 않은 경우 S3 Transfer Manager는 SDK for Java 2.x에서 사용되는 표준 S3 비동기식 클라이언트를 기반으로 구축됩니다.

S3 클라이언트의 사용자 지정 구성 - 멀티파트 사용 필요

```
        S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
                .multipartEnabled(true)
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        S3TransferManager transferManager = S3TransferManager.builder()
                .s3Client(s3AsyncClient)
                .build();
```

S3 클라이언트 구성 없음 - 멀티파트 지원이 자동으로 사용 설정됨

```
S3TransferManager transferManager = S3TransferManager.create();
```

S3 버킷으로 파일을 업로드하려면

다음 예제는 업로드 진행 상황을 기록하는 LoggingTransferListener의 선택적 사용과 함께 파일 업로드 예제를 보여줍니다.

S3 Transfer Manager를 사용하여 Amazon S3에 파일을 업로드하려면 `UploadFileRequest`객체를 `S3TransferManager`의 uploadFile 메서드에 전달하세요.

`uploadFile` 메서드에서 반환된 FileUpload 객체는 업로드 프로세스를 나타냅니다. 요청이 완료되면 CompletedFileUpload 객체에는 업로드에 대한 정보가 포함됩니다.

```
    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!
     */
   }
```

가져오기

```
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;
```

S3 버킷에서 파일 다운로드

다음 예시는 다운로드 예시와 함께 다운로드 진행 상황을 기록하는 LoggingTransferListener의 선택적 사용을 보여줍니다.

S3 Transfer Manager를 사용하여 S3 버킷에서 객체를 다운로드하려면 DownloadFileRequest 객체를 만들어 downloadFile 메서드에 전달합니다.

S3TransferManager의 downloadFile 메서드에서 반환된 FileDownload 객체는 파일 전송을 나타냅니다. 다운로드가 완료되면 CompletedFileDownload에는 다운로드에 대한 정보에 대한 액세스 권한이 포함됩니다.

```
   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!
       */
   }
```

가져오기

```
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.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.NoSuchFileException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;
```

다른 버킷에 Amazon S3 객체를 추가

다음 예는 S3 Transfer Manager로 객체를 복사하는 방법을 보여줍니다.

S3 버킷에서 다른 버킷으로 객체 복사를 시작하려면 기본 CopyObjectRequest 인스턴스를 생성하세요.

다음으로, 기본 `CopyObjectRequest` 항목을 S3 Transfer Manager가사용할 수 있는 CopyRequest로 래핑합니다.

`S3TransferManager`의 `copy` 메서드에서 반환된 `Copy` 객체는 복사 프로세스를 나타냅니다. 복사 프로세스가 완료되면 CompletedCopy 객체에는 응답에 대한 세부 정보가 포함됩니다.

```
    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**
>
> S3 Transfer Manager로 교차 리전 복사를 수행하려면 다음 코드 조각에 표시된 대로 AWS CRT 기반 S3 클라이언트 빌더에서 `crossRegionAccessEnabled`를 활성화합니다.
>
> ```
> S3AsyncClient s3AsyncClient = S3AsyncClient.crtBuilder()
>                 .crossRegionAccessEnabled(true)
>                 .build();
>
> S3TransferManager transferManager = S3TransferManager.builder()
>                 .s3Client(s3AsyncClient)
>                 .build();
> ```

가져오기

```
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;
```

S3 버킷에 로컬 디렉토리 업로드

다음 예는 로컬 디렉터리를 S3에 업로드하는 방법을 보여줍니다.

먼저 `S3TransferManager` 인스턴스의 uploadDirectory 메서드를 호출하여 UploadDirectoryRequest를 전달합니다.

DirectoryUpload 객체는 업로드 프로세스를 나타내며, 요청이 완료되면 CompletedDirectoryUpload를 생성합니다. `CompleteDirectoryUpload` 객체에는 전송에 실패한 파일을 포함하여 전송 결과에 대한 정보가 들어 있습니다.

```
   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();
   }
```

가져오기

```
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;
```

로컬 디렉터리로 S3 버킷 객체 다운로드

다음 예시와 같이 S3 버킷에 있는 객체를 로컬 디렉터리로 다운로드할 수 있습니다.

S3 버킷의 객체를 로컬 디렉토리로 다운로드하려면 먼저 Transfer Manager의 DownloadDirectory 메서드를 호출하고 DownloadDirectoryRequest를 전달하세요.

DirectoryDownload 객체는 업로드 프로세스를 나타내며, 요청이 완료되면 CompletedDirectoryDownload를 생성합니다. CompleteDirectoryDownload 객체에는 전송에 실패한 파일을 포함하여 전송 결과에 대한 정보가 들어 있습니다.

```
    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();
    }
```

가져오기

```
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;
```

전체 예제 보기

이 페이지의 모든 예제에 대한 전체 코드는 GitHub에 포함되어 있습니다.

S3 이벤트 알림 작업

버킷의 활동을 모니터링하는 데 도움이 되도록 Amazon S3는 특정 이벤트가 발생할 때 알림을 보낼 수 있습니다. Amazon S3 사용 설명서에서는 버킷이 보낼 수 있는 알림에 대한 정보를 얻을 수 있습니다.

SDK for Java를 사용하여 4개의 가능한 대상으로 이벤트를 보내도록 버킷을 설정할 수 있습니다.

- Amazon Simple Notification Service(Amazon SNS) 주제
- Amazon Simple Queue Service 대기열
- AWS Lambda 함수
- Amazon EventBridge

EventBridge로 이벤트를 전송하도록 버킷을 설정할 때 동일한 이벤트를 여러 대상으로 팬아웃하도록 EventBridge 규칙을 구성할 수 있습니다. 처음 3개 대상 중 하나로 직접 전송하도록 버킷을 구성하는 경우 각 이벤트에 하나의 대상 유형만 지정 가능합니다.

다음 섹션에서는 SDK for Java를 사용하여 Amazon SQS 대기열과 EventBridge라는 2가지 방법으로 S3 이벤트 알림을 전송하도록 버킷을 구성하는 방법을 알아봅니다.

마지막 섹션에서는 S3 이벤트 알림 API를 사용하여 객체 중심 방식으로 알림을 사용하는 방법을 보여줍니다.

대상으로 직접 전송하도록 버킷 구성

다음 예제에서는 객체 만들기 이벤트 또는 객체 태그 지정 이벤트가 버킷에서 발생할 때 알림을 보내도록 버킷을 구성합니다.

```
static void processS3Events(String bucketName, String queueArn) {
    // Configure the bucket to send Object Created and Object Tagging notifications to
 an existing SQS queue.
    s3Client.putBucketNotificationConfiguration(b -> b
            .notificationConfiguration(ncb -> ncb
                    .queueConfigurations(qcb -> qcb
                            .events(Event.S3_OBJECT_CREATED, Event.S3_OBJECT_TAGGING)
```

```
                            .queueArn(queueArn)))
                    .bucket(bucketName)
    );
}
```

위에 표시된 코드는 2가지 유형의 이벤트를 수신하도록 하나의 대기열을 설정합니다. 편리하게 queueConfigurations 메서드를 사용하면 필요한 경우 여러 대기열 대상을 설정할 수 있습니다. 또한 notificationConfiguration 메서드에서 하나 이상의 Amazon SNS 주제 또는 하나 이상의 Lambda 함수와 같은 추가 대상을 설정할 수 있습니다. 다음 코드 조각은 대기열 2개와 대상 유형 3개가 있는 예제를 보여줍니다.

```
s3Client.putBucketNotificationConfiguration(b -> b
                .notificationConfiguration(ncb -> ncb
                        .queueConfigurations(qcb -> qcb
                                .events(Event.S3_OBJECT_CREATED,
 Event.S3_OBJECT_TAGGING)
                                .queueArn(queueArn),
                                qcb2 -> qcb2.<...>)
                        .topicConfigurations(tcb -> tcb.<...>)
                        .lambdaFunctionConfigurations(lfcb -> lfcb.<...>))
                        .bucket(bucketName)
        );
```

코드 예제 GitHub 리포지토리에는 S3 이벤트 알림을 대기열로 직접 전송하는 전체 예제가 포함되어 있습니다.

EventBridge로 전송할 버킷 구성

다음 예시에서는 EventBridge에 알림을 보내도록 버킷을 구성합니다.

```
public static String setBucketNotificationToEventBridge(String bucketName) {
    // Enable bucket to emit S3 Event notifications to EventBridge.
    s3Client.putBucketNotificationConfiguration(b -> b
            .bucket(bucketName)
            .notificationConfiguration(b1 -> b1
                    .eventBridgeConfiguration(SdkBuilder::build))
    .build());
```

EventBridge로 이벤트를 보내도록 버킷을 구성할 때 EventBridge가 디스패치할 이벤트 유형이나 최종 대상이 아닌 EventBridge 대상을 나타내기만 하면 됩니다. Java SDK의 EventBridge 클라이언트를 사용하여 최종 대상 및 이벤트 유형을 구성합니다.

다음 코드는 객체 만들기 이벤트를 주제 및 대기열로 팬아웃하도록 EventBridge를 구성하는 방법을 보여줍니다.

```
  public static String configureEventBridge(String topicArn, String queueArn) {
       try {
           // Create an EventBridge rule to route Object Created notifications.
           PutRuleRequest putRuleRequest = PutRuleRequest.builder()
                   .name(RULE_NAME)
                   .eventPattern("""
                           {
                             "source": ["aws.s3"],
                             "detail-type": ["Object Created"],
                             "detail": {
                               "bucket": {
                                 "name": ["%s"]
                               }
                             }
                           }
                           """.formatted(bucketName))
                   .build();

           // Add the rule to the default event bus.
           PutRuleResponse putRuleResponse = eventBridgeClient.putRule(putRuleRequest)
                   .whenComplete((r, t) -> {
                       if (t != null) {
                           logger.error("Error creating event bus rule: " +
t.getMessage(), t);
                           throw new RuntimeException(t.getCause().getMessage(), t);
                       }
                       logger.info("Event bus rule creation request sent successfully.
ARN is: {}", r.ruleArn());
                   }).join();

           // Add the existing SNS topic and SQS queue as targets to the rule.
           eventBridgeClient.putTargets(b -> b
                   .eventBusName("default")
                   .rule(RULE_NAME)
                   .targets(List.of (
                           Target.builder()
                                   .arn(queueArn)
                                   .id("Queue")
                                   .build(),
                           Target.builder()
```

```
                                .arn(topicArn)
                                .id("Topic")
                                .build())
                        )
                ).join();
            return putRuleResponse.ruleArn();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
```

Java 코드에서 EventBridge로 작업하려면 Maven `pom.xml` 파일에 `eventbridge` 아티팩트에 대한 종속성을 추가합니다.

```
<dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>eventbridge</artifactId>
</dependency>
```

코드 예제 GitHub 리포지토리에는 S3 이벤트 알림을 EventBridge로 전송한 다음 주제 및 대기열로 전송하는 전체 예제가 포함되어 있습니다.

S3 이벤트 알림 API를 사용하여 이벤트 처리

대상이 S3 알림 이벤트를 수신한 후 S3 이벤트 알림 API를 사용하여 객체 중심 방식으로 이를 처리할 수 있습니다. S3 이벤트 알림 API를 사용하여 대상으로 직접 디스패치되는 이벤트 알림(첫 번째 예제 참조)을 사용할 수 있지만 EventBridge를 통해 라우팅되는 알림은 사용할 수 없습니다. 버킷에서 EventBridge로 전송된 S3 이벤트 알림에는 S3 이벤트 알림 API가 현재 처리하지 않는 다른 구조가 포함되어 있습니다.

종속성 추가

S3 이벤트 알림 API는 SDK for Java 2.x의 버전 2.25.11과 함께 릴리스되었습니다.

S3 이벤트 알림 API를 사용하려면 다음 코드 조각과 같이 Maven `pom.xml`에 필요한 종속성 요소를 추가합니다.

```
<dependencyManagement>
    <dependencies>
```

```
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>bom</artifactId>
            <version>2.X.X[1]</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
<dependencies>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>s3-event-notifications</artifactId>
    </dependency>
</dependencies>
```

[1] 최신 버전.

S3EventNotification 클래스 사용

JSON 문자열에서 **S3EventNotification** 인스턴스 만들기

JSON 문자열을 S3EventNotification 객체로 변환하려면 다음 예제와 같이 S3EventNotification 클래스의 정적 메서드를 사용합니다.

```
import software.amazon.awssdk.eventnotifications.s3.model.S3EventNotification
import software.amazon.awssdk.eventnotifications.s3.model.S3EventNotificationRecord
import software.amazon.awssdk.services.sqs.model.Message;

public class S3EventNotificationExample {
    ...

    void receiveMessage(Message message) {
       // Message received from SQSClient.
       String sqsEventBody = message.body();
       S3EventNotification s3EventNotification =
 S3EventNotification.fromJson(sqsEventBody);

       // Use getRecords() to access all the records in the notification.


       List<S3EventNotificationRecord> records = s3EventNotification.getRecords();
```

```
        S3EventNotificationRecord record = records.stream().findFirst();
        // Use getters on the record to access individual attributes.
        String awsRegion = record.getAwsRegion();
        String eventName = record.getEventName();
        String eventSource = record.getEventSource();

    }
}
```

이 예제에서 `fromJson` 메서드는 JSON 문자열을 `S3EventNotification` 객체로 변환합니다. JSON 문자열에 누락된 필드가 있으면 해당 Java 객체 필드에 `null` 값이 표시되고 JSON의 추가 필드는 무시됩니다.

이벤트 알림 레코드에 대한 다른 API는 `S3EventNotificationRecord`에 대한 API 참조에서 확인할 수 있습니다.

`S3EventNotification` 인스턴스를 JSON 문자열로 변환

다음 예제와 같이 `toJson`(또는 `toJsonPretty`) 메서드를 사용하여 `S3EventNotification` 객체를 JSON 문자열로 변환합니다.

```
import software.amazon.awssdk.eventnotifications.s3.model.S3EventNotification

public class S3EventNotificationExample {
    ...

    void toJsonString(S3EventNotification event) {

        String json = event.toJson();
        String jsonPretty = event.toJsonPretty();

        System.out.println("JSON: " + json);
        System.out.println("Pretty JSON: " + jsonPretty);
    }
}
```

`GlacierEventData`, `ReplicationEventData`, `IntelligentTieringEventData`, `LifecycleEventData`에 대한 필드는 `null`인 경우 JSON에서 제외됩니다. 다른 `null` 필드는 `null`로 직렬화됩니다.

다음은 S3 객체 태그 지정 이벤트에 대한 `toJsonPretty` 메서드의 출력 예제입니다.

```
{
  "Records" : [ {
    "eventVersion" : "2.3",
    "eventSource" : "aws:s3",
    "awsRegion" : "us-east-1",
    "eventTime" : "2024-07-19T20:09:18.551Z",
    "eventName" : "ObjectTagging:Put",
    "userIdentity" : {
      "principalId" : "AWS:XXXXXXXXXXX"
    },
    "requestParameters" : {
      "sourceIPAddress" : "XXX.XX.XX.XX"
    },
    "responseElements" : {
      "x-amz-request-id" : "XXXXXXXXXXXX",
      "x-amz-id-2" : "XXXXXXXXXXXXX"
    },
    "s3" : {
      "s3SchemaVersion" : "1.0",
      "configurationId" : "XXXXXXXXXXXXX",
      "bucket" : {
        "name" : "amzn-s3-demo-bucket",
        "ownerIdentity" : {
          "principalId" : "XXXXXXXXXXX"
        },
        "arn" : "arn:aws:s3:::XXXXXXXXXX"
      },
      "object" : {
        "key" : "akey",
        "size" : null,
        "eTag" : "XXXXXXXXXX",
        "versionId" : null,
        "sequencer" : null
      }
    }
  } ]
}
```

API를 사용하여 Amazon SQS 대기열에서 수신한 알림으로 작업하는 방법을 보여주는 전체 예제는 GitHub에서 사용할 수 있습니다.

Java 라이브러리를 사용하여 Lambda에서 S3 이벤트 처리: AWS SDK for Java 2.x 및 `aws-lambda-java-events`

SDK for Java 2.x를 사용하여 Lambda 함수에서 Amazon S3 이벤트 알림을 처리하는 대신 버전 3.x.x의 `aws-lambda-java-events` 라이브러리를 사용할 수 있습니다. AWS는 `aws-lambda-java-events` 라이브러리를 독립적으로 유지하며 자체 종속성 요구 사항을 보유하고 있습니다. `aws-lambda-java-events` 라이브러리는 Lambda 함수의 S3 이벤트에서만 작동하는 반면, SDK for Java 2.x는 Lambda 함수, Amazon SNS 및 Amazon SQS의 S3 이벤트에서 작동합니다.

두 접근 방식 모두 유사한 API를 사용하여 객체 중심 방식으로 JSON 이벤트 알림 페이로드를 모델링합니다. 다음 표에는 두 접근 방식을 사용하는 데 있어 주목할 만한 차이점이 나와 있습니다.

	AWS SDK for Java	aws-lambda-java-events 라이브러리
패키지 이름 지정	`software.amazon.awssdk.eventnotifications.s3.model.S3EventNotification`	`com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification`

`RequestHandler` 파라미터	Lambda 함수의 `RequestHandler` 구현을 작성하여 JSON 문자열을 수신합니다.
	`import com.amazonaws.services.lambda.runtime.Context;`
	`import com.amazonaws.services.lambda.runtime.RequestHandler;`
	`import software.amazon.awssdk.eventnotifications.s3.model.S3EventNotification;`
	`public class Handler implements RequestHa`
		Lambda 함수의 `RequestHandler` 구현을 작성하여 `S3Event` 객체를 수신합니다.
		`import com.amazonaws.services.lambda.runtime.Context;`
		`import com.amazonaws.services.lambda.runtime.RequestHandler;`
		`import com.amazonaws.services.lambda.runtime.events.S3Event;`
		`public class Handler implements RequestHa`

	AWS SDK for Java	aws-lambda-java-events 라이브러리

	ndler<String, String>
	{
	
	@Override
	public String
	handleRequest(String
	jsonS3Event, Context
	context) {
	S3EventNo
	tification s3Event =
	S3EventNotification
	
	.fromJson(jsonS3Ev
	ent);
	// Work with
	the s3Event object.
	
	...
	}
	}
		ndler<S3Event, String>
		{
		
		@Override
		public String
		handleRequest(S3Ev
		ent s3event, Context
		context) {
		// Work with
		the s3Event object.
		
		...
		}
		}

	AWS SDK for Java	aws-lambda-java-events 라이브러리
Maven 종속성	<dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> 2.X.X</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifactId>s3- event-notifications</ artifactId> </dependency> <!-- Add other SDK dependencies that you need. --> </dependencies>	<dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> 2.X.X</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <!-- The following two dependencies are for the aws-lambda- java-events library. -- > <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-lambda-java- core</artifactId> <version> 1.2.3</version> </dependency> <dependency> <groupId> com.amazonaws</gro upId>

	AWS SDK for Java	aws-lambda-java-events 라이브러리

```
        <artifact
Id>aws-lambda-java-
events</artifactId>
        <version>
3.15.0</version>
    </dependency>
    <!-- Add other SDK
 dependencies that you
 need. -->
</dependencies>
```

Amazon Simple Notification Service 작업

Amazon Simple Notification Service를 사용하면 여러 통신 채널을 통해 애플리케이션의 실시간 알림 메시지를 구독자에게 쉽게 푸시할 수 있습니다. 이 항목에서는 Amazon SNS의 기본 기능을 수행하는 방법을 설명합니다.

주제 생성

주제는 메시지를 전송할 시스템을 정의하는 통신 채널의 논리적 그룹화입니다(예: 메시지를 AWS Lambda 및 HTTP Webhook에 팬 아웃). 메시지를 Amazon SNS에 전송하면 메시지는 주제에 정의된 채널로 배포됩니다. 이렇게 하면 구독자가 메시지를 사용할 수 있게 됩니다.

주제를 생성하려면 먼저 빌더의 `name()` 메서드를 사용하여 세트 이름을 설정하여 CreateTopicRequest 객체를 빌드합니다. 그런 다음 SnsClient의 `createTopic()` 메서드를 사용하여 요청 객체를 Amazon SNS에 전송합니다. 다음 코드 조각과 같이, 이 요청의 결과를 CreateTopicResponse 객체로 캡처할 수 있습니다.

가져옵니다.

```
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;
```

코드

```
    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 "";
    }
```

GitHub의 전체 예제를 참조하세요.

Amazon SNS 주제 나열

기존 Amazon SNS 주제의 목록을 검색하려면 ListTopicsRequest 객체를 빌드합니다. 그런 다음 Amazon SNS의 `listTopics()` 메서드를 사용하여 요청 객체를 `SnsClient`에 전송합니다. 이 요청의 결과를 ListTopicsResponse 객체로 캡처할 수 있습니다.

다음 코드 조각은 요청의 HTTP 상태 코드와 Amazon SNS 주제의 Amazon 리소스 이름(ARN) 목록을 인쇄합니다.

가져옵니다.

```
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;
```

코드

```
    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);
        }
    }
```

GitHub의 전체 예제를 참조하세요.

주제에 엔드포인트 구독 설정

주제를 생성한 후 해당 주제의 엔드포인트가 될 통신 채널을 구성할 수 있습니다. Amazon SNS가 메시지를 수신한 후 메시지는 이러한 엔드포인트에 배포됩니다.

통신 채널을 주제의 엔드포인트로 구성하려면 해당 엔드포인트가 주제에 구독하도록 설정합니다. 시작하려면 SubscribeRequest 객체를 빌드합니다. 통신 채널(예: `lambda` 또는 `email`)을 `protocol()`로 지정합니다. `endpoint()`을 관련 출력 위치(예: Lambda 함수의 ARN 또는 이메일 주소)로 설정한 다음, 구독하려는 주제의 ARN을 `topicArn()`으로 설정합니다. 그런 다음 `SnsClient`의 `subscribe()` 메서드를 사용하여 요청 객체를 Amazon SNS에 전송합니다. 이 요청의 결과를 SubscribeResponse 객체로 캡처할 수 있습니다.

다음 코드 조각은 전자 메일 주소가 주제를 구독하도록 설정하는 방법을 보여 줍니다.

가져옵니다.

```
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;
```

코드

```
    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);
        }
    }
```

GitHub의 전체 예제를 참조하세요.

주제에 메시지 게시

주제를 생성하고 주제에 대해 하나 이상의 엔드포인트를 구성한 후에는 메시지를 주제에 게시할 수 있습니다. 시작하려면 PublishRequest 객체를 빌드합니다. 전송할 `message()`와 메시지를 전송할 주제의 ARN(`topicArn()`)을 지정합니다. 그런 다음 Amazon SNS의 `publish()` 메서드를 사용하여 요청 객체를 `SnsClient`에 전송합니다. 이 요청의 결과를 PublishResponse 객체로 캡처할 수 있습니다.

가져옵니다.

```
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;
```

코드

```
    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);
        }
    }
```

GitHub의 전체 예제를 참조하세요.

주제에서 엔드포인트 구독 취소

주제의 엔드포인트로 구성된 통신 채널을 제거할 수 있습니다. 이렇게 한 후에도 주제 자체는 계속 존재하며 해당 주제에 대해 구성된 다른 엔드포인트에 메시지를 배포합니다.

주제의 엔드포인트로 구성된 통신 채널을 제거하려면 주제에서 해당 엔드포인트의 구독을 해지합니다. 시작하려면 UnsubscribeRequest 객체를 빌드하고 구독 해지할 주제의 ARN을 `subscriptionArn()`으로 설정합니다. 그런 다음 `unsubscribe()`의 `SnsClient` 메서드를 사용하여 요청 객체를 SNS에 전송합니다. 이 요청의 결과를 UnsubscribeResponse 객체로 캡처할 수 있습니다.

가져옵니다.

```
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;
```

코드

```
    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);
        }
    }
```

GitHub의 전체 예제를 참조하세요.

주제 삭제

Amazon SNS 주제를 삭제하려면 먼저 빌더에서 주제 세트의 ARN을 `topicArn()` 메서드로 사용하여 DeleteTopicRequest 객체를 빌드합니다. 그런 다음 Amazon SNS의 `deleteTopic()` 메서드를 사용하여 요청 객체를 `SnsClient`에 전송합니다. 다음 코드 조각과 같이, 이 요청의 결과를 DeleteTopicResponse 객체로 캡처할 수 있습니다.

가져옵니다.

```
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;
```

코드

```
    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);
        }
    }
```

GitHub의 전체 예제를 참조하세요.

자세한 내용은 Amazon Simple Notification Service 개발자 안내서를 참조하세요.

Amazon Simple Queue Service 작업

이 단원에서는 AWS SDK for Java 2.x를 사용한 Amazon Simple Queue Service 프로그래밍의 예제를 제공합니다.

다음 예제에는 각 기술을 보여주는 데 필요한 코드만 포함되어 있습니다. 전체 예제 코드는 GitHub에 있습니다. 이 위치에서 단일 소스 파일을 다운로드하거나 리포지토리를 로컬로 복사하여 모든 예제를 빌드하고 실행할 수 있습니다.

주제

- AWS SDK for Java 2.x를 통해 Amazon SQS에서 자동 요청 배치 처리 사용
- Amazon Simple Queue Service 메시지 대기열 작업
- Amazon Simple Queue Service 메시지를 전송, 수신 및 삭제

AWS SDK for Java 2.x를 통해 Amazon SQS에서 자동 요청 배치 처리 사용

Amazon SQS용 Automatic Request Batching API는 SQS 작업에 대한 요청을 배치 처리 및 버퍼링하는 효율적인 방법을 제공하는 개괄적인 라이브러리입니다. 배치 처리 API를 사용하면 SQS에 대한 요청 수를 줄여 처리량을 개선하고 비용을 최소화할 수 있습니다.

배치 API 메서드는 `SqsAsyncClient` 메서드(`sendMessage`, `changeMessageVisibility`, `deleteMessage`, `receiveMessage`)와 일치하므로 최소한의 변경 사항으로 배치 API를 드롭인 대체로 사용할 수 있습니다.

이 주제에서는 Amazon SQS용 Automatic Request Batching API를 구성하고 사용하는 방법에 대한 개요를 제공합니다.

사전 조건 확인

배치 처리 API에 액세스하려면 SDK for Java 2.x의 버전 2.28.0 이상을 사용해야 합니다. Maven `pom.xml`에는 최소한 다음 요소가 포함되어야 합니다.

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>bom</artifactId>
            <version>2.28.23[1]</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
<dependencies>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>sqs</artifactId>
    </dependency>
</dependencies>
```

[1] 최신 버전

배치 관리자 만들기

자동 요청 배치 처리 API는 SqsAsyncBatchManager 인터페이스에 따라 구현됩니다. 몇 가지 방법으로 관리자의 인스턴스를 만들 수 있습니다.

SqsAsyncClient를 사용한 기본 구성

배치 관리자를 만드는 가장 간단한 방법은 기존 SqsAsyncClient 인스턴스에서 `batchManager` 팩토리 메서드를 호출하는 것입니다. 이 접근 방법은 다음 코드 조각에 나와 있습니다.

```
SqsAsyncClient asyncClient = SqsAsyncClient.create();
SqsAsyncBatchManager sqsAsyncBatchManager = asyncClient.batchManager();
```

이 접근 방식을 사용할 경우 `SqsAsyncBatchManager` 인스턴스는 the section called “구성 설정” 섹션의 표에 표시된 기본값을 사용합니다. 또한 `SqsAsyncBatchManager` 인스턴스는 만들어진 `SqsAsyncClient` 인스턴스의 `ExecutorService`를 사용합니다.

SqsAsyncBatchManager.Builder를 사용한 사용자 지정 구성

고급 사용 사례의 경우 SqsAsyncBatchManager.Builder를 사용하여 배치 관리자를 사용자 지정할 수 있습니다. 이 접근 방식을 사용하여 SqsAsyncBatchManager 인스턴스를 만들면 배치 처리 동작을 미세 조정할 수 있습니다. 다음 코드 조각은 빌더를 사용하여 배치 처리 동작을 사용자 지정하는 방법의 예를 보여줍니다.

```
SqsAsyncBatchManager batchManager = SqsAsyncBatchManager.builder()
    .client(SqsAsyncClient.create())
    .scheduledExecutor(Executors.newScheduledThreadPool(5))
    .overrideConfiguration(b -> b
        .receiveMessageMinWaitDuration(Duration.ofSeconds(10))
        .receiveMessageVisibilityTimeout(Duration.ofSeconds(1))
        .receiveMessageAttributeNames(Collections.singletonList("*"))

 .receiveMessageSystemAttributeNames(Collections.singletonList(MessageSystemAttributeName.ALL))
    .build();
```

이 접근 방식을 사용하면 the section called “구성 설정” 섹션의 표에 표시된 BatchOverrideConfiguration 객체의 설정을 조정할 수 있습니다. 이 접근 방식을 사용하여 배치 관리자에 사용자 지정 ScheduledExecutorService를 제공할 수도 있습니다.

메시지 전송

배치 관리자로 메시지를 보내려면 SqsAsyncBatchManager#sendMessage 메서드를 사용합니다. SDK는 요청을 버퍼링하고 maxBatchSize 또는 sendRequestFrequency 값에 도달하면 요청을 배치로 보냅니다.

다음 예제에서는 다른 요청 바로 다음에 오는 sendMessage 요청을 보여줍니다. 이 경우 SDK는 두 메시지를 단일 배치로 전송합니다.

```
// Sending the first message
CompletableFuture<SendMessageResponse> futureOne =
    sqsAsyncBatchManager.sendMessage(r -> r.messageBody("One").queueUrl("queue"));

// Sending the second message
CompletableFuture<SendMessageResponse> futureTwo =
    sqsAsyncBatchManager.sendMessage(r -> r.messageBody("Two").queueUrl("queue"));

// Waiting for both futures to complete and retrieving the responses
SendMessageResponse messageOne = futureOne.join();
```

```
SendMessageResponse messageTwo = futureTwo.join();
```

메시지 표시 제한 시간 변경

SqsAsyncBatchManager#changeMessageVisibility 메서드를 사용하여 배치 단위로 메시지의 표시 제한 시간을 변경할 수 있습니다. SDK는 요청을 버퍼링하고 maxBatchSize 또는 sendRequestFrequency 값에 도달하면 요청을 배치로 보냅니다.

다음 예제에는 changeMessageVisibility 메서드를 호출하는 방법이 나와 있습니다.

```
CompletableFuture<ChangeMessageVisibilityResponse> futureOne =
    sqsAsyncBatchManager.changeMessageVisibility(r ->
        r.receiptHandle("receiptHandle")
         .queueUrl("queue"));
ChangeMessageVisibilityResponse response = futureOne.join();
```

메시지 삭제

SqsAsyncBatchManager#deleteMessage 메서드를 사용하여 배치 단위로 메시지를 삭제할 수 있습니다. SDK는 요청을 버퍼링하고 maxBatchSize 또는 sendRequestFrequency 값에 도달하면 요청을 배치로 보냅니다.

다음 예제에서는 deleteMessage 메서드를 호출하는 방법을 보여줍니다.

```
CompletableFuture<DeleteMessageResponse> futureOne =
    sqsAsyncBatchManager.deleteMessage(r ->
        r.receiptHandle("receiptHandle")
         .queueUrl("queue"));
DeleteMessageResponse response = futureOne.join();
```

메시지 수신

기본 설정 사용

애플리케이션에서 SqsAsyncBatchManager#receiveMessage 메서드를 폴링하면 배치 관리자가 내부 버퍼에서 메시지를 가져와 SDK가 백그라운드에서 자동으로 업데이트됩니다.

다음 예제에는 receiveMessage 메서드를 호출하는 방법이 나와 있습니다.

```
CompletableFuture<ReceiveMessageResponse> responseFuture =
```

```
    sqsAsyncBatchManager.receiveMessage(r -> r.queueUrl("queueUrl"));
```

사용자 지정 설정 사용

예를 들어 사용자 지정 대기 시간을 설정하고 검색할 메시지 수를 지정하여 요청을 추가로 사용자 지정하려면 다음 예제와 같이 요청을 사용자 지정할 수 있습니다.

```
CompletableFuture<ReceiveMessageResponse> response =
    sqsAsyncBatchManager.receiveMessage(r ->
        r.queueUrl("queueUrl")
         .waitTimeSeconds(5)
         .visibilityTimeout(20));
```

> **Note**
>
> 다음 파라미터가 포함된 ReceiveMessageRequest를 사용하여 receiveMessage를 호출하면 SDK는 배치 관리자를 우회하고 일반 비동기식 receiveMessage 요청을 보냅니다.
>
> - messageAttributeNames
> - messageSystemAttributeNames
> - messageSystemAttributeNamesWithStrings
> - overrideConfiguration

SqsAsyncBatchManager에 대한 구성 설정 재정의

SqsAsyncBatchManager 인스턴스를 만들 때 다음 설정을 조정할 수 있습니다. 다음 설정 목록을 BatchOverrideConfiguration.Builder에서 사용할 수 있습니다.

설정	설명	기본값
maxBatchSize	각 SendMessageBatchRequest , ChangeMessageVisibilityBatchRequest 또는 DeleteMessageBatchRequest 에 대한 배치당 최대 요청 수입니다. 최대값은 10입니다.	10

설정	설명	기본값
`sendRequestFrequency`	배치를 보내기 전까지의 시간이며, `maxBatchSize` 에 도달하면 앞서 전송됩니다. 값이 높을수록 요청이 줄어들지만 지연 시간은 늘어날 수 있습니다.	200ms
`receiveMessageVisibilityTimeout`	메시지에 대한 표시 제한 시간입니다. 설정되지 않으면 대기열의 기본값이 사용됩니다.	대기열의 기본값
`receiveMessageMinWaitDuration`	`receiveMessage` 요청의 최소 대기 시간입니다. CPU 낭비를 방지하기 위해 0으로 설정하는 것은 피합니다.	50ms
`receiveMessageSystemAttributeNames`	`receiveMessage` 호출을 요청할 시스템 속성 이름 목록입니다.	없음
`receiveMessageAttributeNames`	`receiveMessage` 호출을 요청할 속성 이름 목록입니다.	없음

Amazon Simple Queue Service 메시지 대기열 작업

메시지 대기열은 메시지를 안정적으로 보내는 데 사용되는 논리적 컨테이너입니다 Amazon Simple Queue Service. 표준과 선입선출(FIFO), 이렇게 두 가지 유형의 대기열이 있습니다. 대기열과 이러한 유형 간의 차이에 대해 자세히 알아보려면 Amazon Simple Queue Service 개발자 안내서를 참조하세요.

이 주제에서는를 사용하여 Amazon Simple Queue Service 대기열의 URL을 생성, 나열, 삭제 및 가져오는 방법을 설명합니다 AWS SDK for Java.

다음 예제에서 사용되는 `sqsClient` 변수는 다음 코드 조각에서 만들 수 있습니다.

```
SqsClient sqsClient = SqsClient.create();
```

정적 `create()` 메서드를 사용하여 `SqsClient`를 만들면 SDK가 기본 리전 공급자 체인을 사용하여 리전을 구성하고 기본 자격 증명 공급자 체인을 사용하여 자격 증명을 구성합니다.

대기열 생성

`SqsClient`'s `createQueue` 메서드를 사용하고 다음 코드 조각과 같이 대기열 파라미터를 설명하는 `CreateQueueRequest` 객체를 제공합니다.

가져오기

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

코드

```
            CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                .queueName(queueName)
                .build();

            sqsClient.createQueue(createQueueRequest);
```

GitHub의 전체 예제를 참조하십시오.

대기열 나열

계정의 Amazon Simple Queue Service 대기열을 나열하려면 `ListQueuesRequest` 객체를 사용하여 `SqsClient`'s `listQueues` 메서드를 호출합니다.

파라미터를 사용하지 않는 `listQueues` 메서드 형식을 사용하면 서비스에서는 최대 1,000개의 대기열까지 모든 대기열을 반환합니다.

`ListQueuesRequest` 객체에 대기열 이름 접두사를 지정해 다음 코드와 같이 접두사와 일치하는 대기열만 결과로 표시되도록 제한할 수 있습니다.

가져오기

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
```

```
import java.util.List;
```

코드

```
        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);
        }
```

GitHub의 전체 예제를 참조하십시오.

대기열의 URL 가져오기

다음 코드는 `GetQueueUrlRequest` 객체로 `SqsClient`'s `getQueueUrl` 메서드를 직접적으로 호출하여 대기열의 URL을 가져오는 방법을 보여줍니다.

가져오기

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

코드

```
            GetQueueUrlResponse getQueueUrlResponse =

 sqsClient.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
            String queueUrl = getQueueUrlResponse.queueUrl();
            return queueUrl;
```

GitHub의 전체 예제를 참조하십시오.

대기열 삭제

대기열의 URL을 DeleteQueueRequest 객체에 제공합니다. 그런 다음, 다음 코드와 같이 SqsClient's deleteQueue 메서드를 직접적으로 호출하여 대기열을 삭제합니다.

가져오기

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

코드

```
    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);
        }
    }
```

GitHub의 전체 예제를 참조하십시오.

추가 정보

- Amazon Simple Queue Service API 참조의 CreateQueue

- Amazon Simple Queue Service API 참조의 GetQueueUrl
- Amazon Simple Queue Service API 참조의 ListQueues
- Amazon Simple Queue Service API 참조의 DeleteQueue

Amazon Simple Queue Service 메시지를 전송, 수신 및 삭제

메시지는 분산된 구성 요소가 송신 및 수신할 수 있는 데이터 조각입니다. 메시지는 항상 SQS 대기열을 사용하여 전달됩니다.

다음 예제에서 사용되는 `sqsClient` 변수는 다음 코드 조각에서 만들 수 있습니다.

```
SqsClient sqsClient = SqsClient.create();
```

정적 `create()` 메서드를 사용하여 `SqsClient`를 만들면 SDK가 기본 리전 공급자 체인을 사용하여 리전을 구성하고 기본 자격 증명 공급자 체인을 사용하여 자격 증명을 구성합니다.

메시지 전송

SqsClient 클라이언트의 `sendMessage` 메서드를 호출하여 Amazon Simple Queue Service 대기열에 단일 메시지를 추가합니다. 대기열의 URL, 메시지 본문 및 선택적 지연 값(초 단위)이 포함된 SendMessageRequest 객체를 제공합니다.

가져옵니다.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

코드

```
            sqsClient.sendMessage(SendMessageRequest.builder()
                .queueUrl(queueUrl)
                .messageBody("Hello world!")
                .delaySeconds(10)
                .build());

            sqsClient.sendMessage(sendMsgRequest);
```

한 번의 요청에 여러 메시지를 전송

SqsClient의 sendMessageBatch 메서드를 사용하여 단일 요청에서 메시지를 2개 이상 전송합니다. 이 메서드는 전송할 메시지 목록과 대기열 URL이 포함된 SendMessageBatchRequest를 가져옵니다. (각 메시지는 SendMessageBatchRequestEntry임) 또 메시지의 지연 값을 설정해 특정 메시지 전송을 지연시킬 수 있습니다.

가져옵니다.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

코드

```
            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);
```

GitHub의 전체 예제를 참조하십시오.

메시지 검색

SqsClient의 receiveMessage 메서드를 호출하여 현재 대기열에 있는 메시지를 검색합니다. 이 메서드는 대기열 URL이 포함된 ReceiveMessageRequest를 가져옵니다. 또 반환할 메시지의 최대 수를 지정할 수 있습니다. 메시지는 Message 객체의 목록으로 반환됩니다.

가져옵니다.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
```

```
import java.util.List;
```

코드

```
        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;
```

GitHub의 전체 예제를 참조하십시오.

수신 후 메시지 삭제

메시지를 수신하고 콘텐츠를 처리한 후에는 메시지의 수신 핸들과 대기열 URL을 SqsClient's deleteMessage 메서드로 전송하여 대기열에서 메시지를 삭제합니다.

가져옵니다.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

코드

```
        try {
            for (Message message : messages) {
                DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .receiptHandle(message.receiptHandle())
                        .build();
                sqsClient.deleteMessage(deleteMessageRequest);
```

```
        }
```

GitHub의 전체 예제를 참조하십시오.

추가 정보

- Amazon Simple Queue Service 개발자 안내서의 Amazon Simple Queue Service 큐 작동 방식
- Amazon Simple Queue Service API 참조의 SendMessage
- Amazon Simple Queue Service API 참조의 SendMessageBatch
- Amazon Simple Queue Service API 참조의 ReceiveMessage
- Amazon Simple Queue Service API 참조의 DeleteMessage

작업 Amazon Transcribe

다음 예제는 Amazon Transcribe를 사용하여 양방향 스트리밍이 작동하는 방식을 보여 줍니다. 양방향 스트리밍은 서비스로 향하는 데이터 스트림과 실시간으로 다시 수신되는 데이터 스트림이 둘 다 있다는 의미입니다. 이 예제에서는 Amazon Transcribe 스트리밍 트랜스크립션을 사용하여 오디오 스트림을 보내고 트랜스크립션된 텍스트 스트림을 실시간으로 다시 수신합니다.

이 기능에 대한 자세한 내용은 Amazon Transcribe 개발자 안내서의 스트리밍 트랜스크립션을 참조하세요.

사용을 시작하려면 Amazon Transcribe 개발자 안내서의 시작하기를 참조하세요 Amazon Transcribe.

마이크 설정

이 코드는 javax.sound.sampled 패키지를 사용하여 입력 디바이스에서 오디오를 스트리밍합니다.

코드

```
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;
    }
}
```

GitHub의 전체 예제를 참조하세요.

게시자 생성

이 코드는 오디오 스트림의 Amazon Transcribe 오디오 데이터를 게시하는 게시자를 구현합니다.

코드

```
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();
        }
    }
}
```

GitHub의 전체 예제를 참조하세요.

클라이언트 생성 및 스트림 시작

주 메서드에서 요청 객체를 만들고 오디오 입력 스트림을 시작하며 오디오 입력으로 게시자를 인스턴스화합니다.

또한 StartStreamTranscriptionResponseHandler를 생성하여 응답을 처리하는 방법을 지정해야 합니다 Amazon Transcribe.

그런 다음 TranscribeStreamingAsyncClient의 `startStreamTranscription` 메서드를 사용하여 양방향 스트리밍을 시작합니다.

가져오기

```
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;
```

코드

```
   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);
         }
    }
```

GitHub의 전체 예제를 참조하세요.

추가 정보

- Amazon Transcribe 개발자 안내서의 작동 방식.
- Amazon Transcribe 개발자 안내서의 오디오 스트리밍 시작하기를 참조하세요.

Java 2.x용 SDK 코드 예제

이 주제의 코드 예제에서는를 AWS SDK for Java 2.x 와 함께 사용하는 방법을 보여줍니다 AWS.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

일부 서비스에는 서비스와 관련된 라이브러리 또는 함수를 활용하는 방법을 보여주는 추가 예제 범주가 포함되어 있습니다.

서비스

- SDK for Java 2.x를 사용한 ACM 예제
- Java 2.x용 SDK를 사용한 API Gateway 예제
- SDK for Java 2.x를 사용한 Application Auto Scaling 예제
- Java 2.x용 SDK를 사용하는 Application Recovery Controller 예제
- Java 2.x용 SDK를 사용하는 Aurora 예제
- Java 2.x용 SDK를 사용하는 Auto Scaling 예제
- AWS Batch SDK for Java 2.x를 사용한 예제
- Java 2.x용 SDK를 사용하는 Amazon Bedrock 예제
- Java 2.x용 SDK를 사용하는 Amazon Bedrock 런타임 예제
- Java 2.x용 SDK를 사용하는 CloudFront 예제
- Java 2.x용 SDK를 사용하는 CloudWatch 예제
- Java 2.x용 SDK를 사용하는 CloudWatch Events 예제
- Java 2.x용 SDK를 사용하는 CloudWatch Logs 예제
- SDK for Java 2.x를 사용한 Amazon Cognito 자격 증명 예시
- Java 2.x용 SDK를 사용하는 Amazon Cognito 자격 증명 공급자 예제
- Java 2.x용 SDK를 사용하는 Amazon Comprehend 예제
- AWS Control Tower SDK for Java 2.x를 사용한 예제
- SDK for Java 2.x를 사용한 Firehose 예제

- SDK for Java 2.x를 사용한 Amazon DocumentDB 예제
- Java 2.x용 SDK를 사용하는 DynamoDB 예제
- Java 2.x용 SDK를 사용하는 Amazon EC2 예제
- SDK for Java 2.x를 사용한 Amazon ECR 예제
- Java 2.x용 SDK를 사용하는 Amazon ECS 예제
- ELB - SDK for Java 2.x를 사용한 버전 2 예제
- SDK for Java 2.x를 사용한 MediaStore 예시
- AWS Entity Resolution SDK for Java 2.x를 사용한 예제
- Java 2.x용 SDK를 사용하는 OpenSearch Service 예제
- Java 2.x용 SDK를 사용하는 EventBridge 예제
- SDK for Java 2.x를 사용한 EventBridge Scheduler 예제
- Java 2.x용 SDK를 사용하는 Forecast 예제
- SDK for Java 2.x를 사용하는 Amazon Glacier 예제
- AWS Glue SDK for Java 2.x를 사용한 예제
- Java 2.x용 SDK를 사용하는 HealthImaging 예제
- Java 2.x용 SDK를 사용하는 IAM 예제
- SDK for Java 2.x를 사용한 Amazon Inspector 예제
- AWS IoT SDK for Java 2.x를 사용한 예제
- AWS IoT data SDK for Java 2.x를 사용한 예제
- AWS IoT FleetWise SDK for Java 2.x를 사용한 예제
- AWS IoT SiteWise SDK for Java 2.x를 사용한 예제
- Java 2.x용 SDK를 사용하는 Amazon Keyspaces 예제
- Java 2.x용 SDK를 사용하는 Kinesis 예제
- AWS KMS SDK for Java 2.x를 사용한 예제
- Java 2.x용 SDK를 사용하는 Lambda 예제
- SDK for Java 2.x를 사용한 Amazon Lex 예제
- SDK for Java 2.x를 사용한 Amazon Location 예제
- SDK for Java 2.x를 사용한 Location Service Places 예제
- AWS Marketplace SDK for Java 2.x를 사용한 카탈로그 API 예제
- AWS Marketplace SDK for Java 2.x를 사용한 계약 API 예제

- Java 2.x용 SDK를 사용하는 MediaConvert 예제
- Java 2.x용 SDK를 사용하는 Migration Hub 예제
- SDK for Java 2.x를 사용한 Amazon MSK 예제
- SDK for Java 2.x를 사용한 Neptune 예제
- SDK for Java 2.x를 사용한 Partner Central 예제
- Java 2.x용 SDK를 사용하는 Amazon Personalize 예제
- SDK for Java 2.x를 사용하는 Amazon Personalize Events 예제
- Java 2.x용 SDK를 사용하는 Amazon Personalize 런타임 예제
- SDK for Java 2.x를 사용하는 Amazon Pinpoint 예제
- Java 2.x용 SDK를 사용하는 Amazon Pinpoint SMS 및 음성 API 예제
- Java 2.x용 SDK를 사용하는 Amazon Polly 예제
- Java 2.x용 SDK를 사용하는 Amazon RDS 예제
- SDK for Java 2.x를 사용한 Amazon RDS 데이터 서비스 예제
- Java 2.x용 SDK를 사용하는 Amazon Redshift 예제
- Java 2.x용 SDK를 사용하는 Amazon Rekognition 예제
- Java 2.x용 SDK를 사용하는 Route 53 도메인 등록 예제
- Java 2.x용 SDK를 사용하는 Amazon S3 예제
- SDK for Java 2.x를 사용한 Amazon S3 Control 예제
- SDK for Java 2.x를 사용한 S3 디렉터리 버킷 예제
- SDK for Java 2.x를 사용한 SageMaker AI 예제
- Java 2.x용 SDK를 사용하는 Secrets Manager 예제
- Java 2.x용 SDK를 사용하는 SES 예제
- Java 2.x용 SDK를 사용하는 SES API v2 예제
- Java 2.x용 SDK를 사용하는 Amazon SNS 예제
- Java 2.x용 SDK를 사용하는 Amazon SQS 예제
- Java 2.x용 SDK를 사용하는 Step Functions의 예제
- AWS STS SDK for Java 2.x를 사용한 예제
- 지원 SDK for Java 2.x를 사용한 예제
- Java 2.x용 SDK를 사용하는 Secrets Manager 예제
- Java 2.x용 SDK를 사용하는 Amazon Textract 예제

- SDK for Java 2.x를 사용한 Amazon Transcribe 예시
- SDK for Java 2.x를 사용한 Amazon Transcribe 스트리밍 예제
- SDK for Java 2.x를 사용한 Amazon Translate 예제

SDK for Java 2.x를 사용한 ACM 예제

다음 코드 예제에서는 ACM과 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`AddTagsToCertificate`

다음 코드 예시는 `AddTagsToCertificate`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
```

```
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AddTagsToCertificate {

    public static void main(String[] args) {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        addTags(certArn);
    }

    /**
     * Adds tags to a certificate in AWS Certificate Manager (ACM).
     *
     * @param certArn the Amazon Resource Name (ARN) of the certificate to add tags
 to
     */
    public static void addTags(String certArn) {
        AcmClient acmClient = AcmClient.create();
        List<Tag> expectedTags =
 List.of(Tag.builder().key("key").value("value").build());
        AddTagsToCertificateRequest addTagsToCertificateRequest =
 AddTagsToCertificateRequest.builder()
            .certificateArn(certArn)
            .tags(expectedTags)
            .build();

        try {
            acmClient.addTagsToCertificate(addTagsToCertificateRequest);
            System.out.println("Successfully added tags to a certificate");
        } catch (AcmException e) {
            System.out.println(e.getMessage());
```

```
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 AddTagsToCertificate를 참조하세요.

`DeleteCertificate`

다음 코드 예시는 `DeleteCertificate`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteCert {

    public static void main(String[] args) {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }
```

```
        String certArn = args[0];
        deleteCertificate(certArn);
    }

    /**
     * Deletes an SSL/TLS certificate from the AWS Certificate Manager (ACM).
     *
     * @param certArn the Amazon Resource Name (ARN) of the certificate to be
 deleted
     */
    public static void deleteCertificate( String certArn) {
        AcmClient acmClient = AcmClient.create();
        DeleteCertificateRequest request = DeleteCertificateRequest.builder()
            .certificateArn(certArn)
            .build();

        try {
            acmClient.deleteCertificate(request);
            System.out.println("The certificate was deleted");

        } catch (AcmException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteCertificate를 참조하세요.

`DescribeCertificate`

다음 코드 예시는 `DescribeCertificate`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DescribeCert {

    public static void main(String[] args) {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        describeCertificate(certArn);
    }

    /**
     * Describes the details of an SSL/TLS certificate.
     *
     * @param certArn the Amazon Resource Name (ARN) of the certificate to describe
     * @throws AcmException if an error occurs while describing the certificate
     */
    public static void describeCertificate(String certArn) {
        AcmClient acmClient = AcmClient.create();
        DescribeCertificateRequest req = DescribeCertificateRequest.builder()
            .certificateArn(certArn)
            .build();

        try {
            DescribeCertificateResponse response =
 acmClient.describeCertificate(req);
```

```
            // Print the certificate details.
            System.out.println("Certificate ARN: " +
 response.certificate().certificateArn());
            System.out.println("Domain Name: " +
 response.certificate().domainName());
            System.out.println("Issued By: " + response.certificate().issuer());
            System.out.println("Issued On: " + response.certificate().issuedAt());
            System.out.println("Status: " + response.certificate().status());
        } catch (AcmException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeCertificate를 참조하세요.

`ExportCertificate`

다음 코드 예시는 `ExportCertificate`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ExportCertificate {

    public static void main(String[] args) throws Exception {
        final String usage = """
```

```
            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        exportCert(certArn);
    }

    /**
     * Exports an SSL/TLS certificate and its associated private key and certificate
chain from AWS Certificate Manager (ACM).
     *
     * @param certArn The Amazon Resource Name (ARN) of the certificate that you
want to export.
     * @throws IOException If an I/O error occurs while reading the private key
passphrase file or exporting the certificate.
     */
    public static void exportCert(String certArn) throws IOException {
        AcmClient acmClient = AcmClient.create();

        // Initialize a file descriptor for the passphrase file.
        RandomAccessFile filePassphrase = null;
        ByteBuffer bufPassphrase = null;

        // Create a file stream for reading the private key passphrase.
        try {
            filePassphrase = new RandomAccessFile("C:\\AWS\\password.txt", "r");
        } catch (IllegalArgumentException | SecurityException |
FileNotFoundException ex) {
            throw ex;
        }

        // Create a channel to map the file.
        FileChannel channelPassphrase = filePassphrase.getChannel();

        // Map the file to the buffer.
        try {
```

```
            bufPassphrase = channelPassphrase.map(FileChannel.MapMode.READ_ONLY, 0,
 channelPassphrase.size());
            channelPassphrase.close();
            filePassphrase.close();
        } catch (IOException ex) {
            throw ex;
        }

        // Create a request object.
        ExportCertificateRequest req = ExportCertificateRequest.builder()
            .certificateArn(certArn)
            .passphrase(SdkBytes.fromByteBuffer(bufPassphrase))
            .build();

        // Export the certificate.
        ExportCertificateResponse result = null;
        try {
            result = acmClient.exportCertificate(req);
        } catch (InvalidArnException | InvalidTagException |
 ResourceNotFoundException ex) {
            throw ex;
        }

        // Clear the buffer.
        bufPassphrase.clear();

        // Display the certificate and certificate chain.
        String certificate = result.certificate();
        System.out.println(certificate);

        String certificateChain = result.certificateChain();
        System.out.println(certificateChain);

        // This example retrieves but does not display the private key.
        String privateKey = result.privateKey();
        System.out.println("The example is complete");
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ExportCertificate를 참조하세요.

`ImportCertificate`

다음 코드 예시는 `ImportCertificate`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ImportCert {

    public static void main(String[] args) {
        final String usage = """
            Usage: <bucketName> <certificateKey> <privateKeyKey>

            Where:
                bucketName - The name of the S3 bucket containing the certificate
 and private key.
                certificateKey - The object key for the SSL/TLS certificate file in
 S3.
                privateKeyKey - The object key for the private key file in S3.
            """;

        if (args.length != 3) {
            System.out.println(usage);
            return;
        }

        String bucketName = args[0];
        String certificateKey = args[1];
        String privateKeyKey = args[2];
```

```
        String certificateArn = importCertificate(bucketName, certificateKey,
privateKeyKey);
        System.out.println("Certificate imported with ARN: " + certificateArn);
    }

    /**
     * Imports an SSL/TLS certificate and private key from S3 into AWS Certificate
Manager (ACM).
     *
     * @param bucketName     The name of the S3 bucket.
     * @param certificateKey The key for the SSL/TLS certificate file in S3.
     * @param privateKeyKey  The key for the private key file in S3.
     * @return The ARN of the imported certificate.
     */
    public static String importCertificate(String bucketName, String certificateKey,
String privateKeyKey) {
        AcmClient acmClient = AcmClient.create();
        S3Client s3Client = S3Client.create();

        try {
            byte[] certificateBytes = downloadFileFromS3(s3Client, bucketName,
certificateKey);
            byte[] privateKeyBytes = downloadFileFromS3(s3Client, bucketName,
privateKeyKey);

            ImportCertificateRequest request = ImportCertificateRequest.builder()

.certificate(SdkBytes.fromByteBuffer(ByteBuffer.wrap(certificateBytes)))

.privateKey(SdkBytes.fromByteBuffer(ByteBuffer.wrap(privateKeyBytes)))
                    .build();

            ImportCertificateResponse response =
acmClient.importCertificate(request);
            return response.certificateArn();

        } catch (IOException e) {
            System.err.println("Error downloading certificate or private key from
S3: " + e.getMessage());
        } catch (S3Exception e) {
            System.err.println("S3 error: " + e.awsErrorDetails().errorMessage());
        }
        return "";
    }
```

```
    /**
     * Downloads a file from Amazon S3 and returns its contents as a byte array.
     *
     * @param s3Client   The S3 client.
     * @param bucketName The name of the S3 bucket.
     * @param objectKey  The key of the object in S3.
     * @return The file contents as a byte array.
     * @throws IOException If an I/O error occurs.
     */
    private static byte[] downloadFileFromS3(S3Client s3Client, String bucketName,
 String objectKey) throws IOException {
        GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

        try (ResponseInputStream<GetObjectResponse> s3Object =
 s3Client.getObject(getObjectRequest);
             ByteArrayOutputStream byteArrayOutputStream = new
 ByteArrayOutputStream()) {
            IoUtils.copy(s3Object, byteArrayOutputStream);
            return byteArrayOutputStream.toByteArray();
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ImportCertificate를 참조하세요.

`ListCertificates`

다음 코드 예시는 `ListCertificates`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListCerts {
    public static void main(String[] args) {
        listCertificates();
    }

    /**
     * Lists all the certificates managed by AWS Certificate Manager (ACM) that have
 a status of "ISSUED".
     */
    public static void listCertificates() {
        AcmClient acmClient = AcmClient.create();
        try {
            ListCertificatesRequest listRequest = ListCertificatesRequest.builder()
                .certificateStatuses(CertificateStatus.ISSUED)
                .maxItems(100)
                .build();
            ListCertificatesIterable listResponse =
 acmClient.listCertificatesPaginator(listRequest);

            // Print the certificate details using streams
            listResponse.certificateSummaryList().stream()
                .forEach(certificate -> {
                    System.out.println("Certificate ARN: " +
 certificate.certificateArn());
                    System.out.println("Certificate Domain Name: " +
 certificate.domainName());
                    System.out.println("Certificate Status: " +
 certificate.statusAsString());
                    System.out.println("---");
                });

        } catch (AcmException e) {
            System.err.println(e.getMessage());
        }
    }
```

```
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListCertificates를 참조하세요.

`ListTagsForCertificate`

다음 코드 예시는 `ListTagsForCertificate`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListCertTags {

    public static void main(String[] args) {

        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }
```

```
        String certArn = args[0];
        listCertTags(certArn);
    }

    /**
     * Lists the tags associated with an AWS Certificate Manager (ACM) certificate.
     *
     * @param certArn the Amazon Resource Name (ARN) of the ACM certificate
     */
    public static void listCertTags(String certArn) {
        AcmClient acmClient = AcmClient.create();

        ListTagsForCertificateRequest request =
 ListTagsForCertificateRequest.builder()
            .certificateArn(certArn)
            .build();

        ListTagsForCertificateResponse response =
 acmClient.listTagsForCertificate(request);
        List<Tag> tagList = response.tags();
        tagList.forEach(tag -> {
            System.out.println("Key: " + tag.key());
            System.out.println("Value: " + tag.value());
        });
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListTagsForCertificate를 참조하세요.

`RemoveTagsFromCertificate`

다음 코드 예시는 `RemoveTagsFromCertificate`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class RemoveTagsFromCert {

    public static void main(String[] args) {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        removeTags(certArn);
    }

    /**
     * Removes tags from an AWS Certificate Manager (ACM) certificate.
     *
     * @param certArn the Amazon Resource Name (ARN) of the certificate from which
 to remove tags
     */
    public static void removeTags(String certArn) {
        AcmClient acmClient = AcmClient.create();
        List<Tag> expectedTags =
 List.of(Tag.builder().key("key").value("value").build());
        RemoveTagsFromCertificateRequest req =
 RemoveTagsFromCertificateRequest.builder()
            .certificateArn(certArn)
            .tags(expectedTags)
```

```
            .build();

        try {
            acmClient.removeTagsFromCertificate(req);
            System.out.println("Successfully removed tags from the certificate");
        } catch (AcmException e) {
            System.err.println(e.getMessage());
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 RemoveTagsFromCertificate를 참조하세요.

`RenewCertificate`

다음 코드 예시는 `RenewCertificate`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class RenewCert {
    public static void main(String[] args) {
        final String usage = """

            Usage:    <certArn>
```

```
            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        renewCertificate(certArn);
    }

    /**
     * Renews an existing SSL/TLS certificate in AWS Certificate Manager (ACM).
     *
     * @param certArn The Amazon Resource Name (ARN) of the certificate to be
 renewed.
     * @throws AcmException If there is an error renewing the certificate.
     */
    public static void renewCertificate(String certArn) {
        AcmClient acmClient = AcmClient.create();

        RenewCertificateRequest certificateRequest =
 RenewCertificateRequest.builder()
            .certificateArn(certArn)
            .build();

        try {
            acmClient.renewCertificate(certificateRequest);
            System.out.println("The certificate was renewed");
        } catch(AcmException e){
            System.out.println(e.getMessage());
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 RenewCertificate를 참조하세요.

`RequestCertificate`

다음 코드 예시는 `RequestCertificate`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RequestCert {

    public static void main(String[] args) {
        requestCertificate();
    }

    /**
     * Requests a certificate from the AWS Certificate Manager (ACM) service.
     */
    public static void requestCertificate() {
        AcmClient acmClient = AcmClient.create();
        ArrayList<String> san = new ArrayList<>();
        san.add("www.example.com");

        RequestCertificateRequest req = RequestCertificateRequest.builder()
            .domainName("example.com")
            .idempotencyToken("1Aq25pTy")
            .subjectAlternativeNames(san)
            .build();

        try {
            RequestCertificateResponse response = acmClient.requestCertificate(req);
            System.out.println("Cert ARN IS " + response.certificateArn());
        } catch (AcmException e) {
            System.err.println(e.getMessage());
```

```
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 RequestCertificate를 참조하세요.

Java 2.x용 SDK를 사용한 API Gateway 예제

다음 코드 예제에서는 API Gateway와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

AWS 커뮤니티 기여는 여러 팀이 생성하고 유지 관리하는 예입니다 AWS. 피드백을 제공하려면 연결된 리포지토리에 제공된 메커니즘을 사용합니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업
- 시나리오
- AWS 커뮤니티 기여

작업

`CreateDeployment`

다음 코드 예시는 `CreateDeployment`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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 "";
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateDeployment를 참조하세요.

`CreateRestApi`

다음 코드 예시는 `CreateRestApi`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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 "";
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateRestApi를 참조하세요.

`DeleteDeployment`

다음 코드 예시는 `DeleteDeployment`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteDeployment를 참조하세요.

`DeleteRestApi`

다음 코드 예시는 `DeleteRestApi`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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);
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteRestApi를 참조하세요.

시나리오

사진을 관리하기 위한 서버리스 애플리케이션 만들기

다음 코드 예시에서는 사용자가 레이블을 사용하여 사진을 관리할 수 있는 서버리스 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 AWS 커뮤니티의 게시물을 참조하세요.

이 예제에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition

- Amazon S3
- Amazon SNS

API Gateway를 사용하여 Lambda 함수 간접 호출

다음 코드 예제에서는 Amazon API Gateway에서 호출한 AWS Lambda 함수를 생성하는 방법을 보여줍니다.

SDK for Java 2.x

Lambda Java 런타임 API를 사용하여 AWS Lambda 함수를 생성하는 방법을 보여줍니다. 이 예제에서는 다양한 AWS 서비스를 호출하여 특정 사용 사례를 수행합니다. 이 예제에서는 Amazon API Gateway에서 간접 호출한 Lambda 함수를 생성하여 작업 기념일에 대한 Amazon DynamoDB 테이블을 스캔하고 Amazon Simple Notification Service(Amazon SNS)를 사용하여 직원에게 1주년 기념일을 축하하는 문자 메시지를 전송하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

AWS 커뮤니티 기여

서버리스 애플리케이션 빌드 및 테스트

다음 코드 예제에서는 Lambda 및 DynamoDB와 함께 API 게이트웨이를 사용하여 서버리스 애플리케이션을 빌드하고 테스트하는 방법을 보여줍니다.

SDK for Java 2.x

Java SDK를 사용하여 Lambda 및 DynamoDB가 포함된 API 게이트웨이로 구성된 서버리스 애플리케이션을 빌드하고 테스트하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda

SDK for Java 2.x를 사용한 Application Auto Scaling 예제

다음 코드 예제에서는 Application Auto Scaling과 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`DeleteScalingPolicy`

다음 코드 예시는 `DeleteScalingPolicy`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteScalingPolicy를 참조하세요.

`RegisterScalableTarget`

다음 코드 예시는 `RegisterScalableTarget`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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());
            }
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 RegisterScalableTarget을 참조하세요.

Java 2.x용 SDK를 사용하는 Application Recovery Controller 예제

다음 코드 예제에서는 Application Recovery Controller와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`GetRoutingControlState`

다음 코드 예시는 `GetRoutingControlState`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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;
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 GetRoutingControlState를 참조하세요.

`UpdateRoutingControlState`

다음 코드 예시는 `UpdateRoutingControlState`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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;
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateRoutingControlState를 참조하세요.

Java 2.x용 SDK를 사용하는 Aurora 예제

다음 코드 예제에서는 Aurora와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시작하기
- 기본 사항
- 작업
- 시나리오

시작하기

Hello Aurora

다음 코드 예제에서는 Aurora를 사용하여 시작하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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()));
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeDBClusters를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 사용자 지정 Aurora DB 클러스터 파라미터 그룹을 만들고 파라미터 값을 설정합니다.
- 파라미터 그룹을 사용하는 DB 클러스터를 생성합니다.
- 데이터베이스가 포함된 DB 인스턴스를 생성합니다.
- DB 클러스터의 스냅샷을 만든 다음, 리소스를 정리합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including 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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 항목을 참조하세요.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots
  - DescribeDBClusters
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

작업

`CreateDBCluster`

다음 코드 예시는 `CreateDBCluster`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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 "";
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateDBCluster를 참조하세요.

`CreateDBClusterParameterGroup`

다음 코드 예시는 `CreateDBClusterParameterGroup`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x SDK for Rust API 참조의 CreateDBClusterParameterGroup를 참조하세요.

`CreateDBClusterSnapshot`

다음 코드 예시는 `CreateDBClusterSnapshot`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateDBClusterSnapshot을 참조하세요.

`CreateDBInstance`

다음 코드 예시는 `CreateDBInstance`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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 "";
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateDBInstance를 참조하세요.

`DeleteDBCluster`

다음 코드 예시는 `DeleteDBCluster`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteDBCluster를 참조하세요.

`DeleteDBClusterParameterGroup`

다음 코드 예시는 `DeleteDBClusterParameterGroup`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteDBClusterParameterGroup을 참조하세요.

`DeleteDBInstance`

다음 코드 예시는 `DeleteDBInstance`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteDBInstance를 참조하세요.

`DescribeDBClusterParameterGroups`

다음 코드 예시는 `DescribeDBClusterParameterGroups`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeDBClusterParameterGroups를 참조하세요.

`DescribeDBClusterParameters`

다음 코드 예시는 `DescribeDBClusterParameters`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeDBClusterParameters를 참조하세요.

`DescribeDBClusterSnapshots`

다음 코드 예시는 `DescribeDBClusterSnapshots`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeDBClusterSnapshots를 참조하세요.

DescribeDBClusters

다음 코드 예시는 DescribeDBClusters의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeDBClusters를 참조하세요.

`DescribeDBEngineVersions`

다음 코드 예시는 `DescribeDBEngineVersions`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeDBEngineVersions을 참조하세요.

`DescribeDBInstances`

다음 코드 예시는 `DescribeDBInstances`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeDBInstances를 참조하세요.

`DescribeOrderableDBInstanceOptions`

다음 코드 예시는 `DescribeOrderableDBInstanceOptions`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeOrderableDBInstanceOptions를 참조하세요.

`ModifyDBClusterParameterGroup`

다음 코드 예시는 `ModifyDBClusterParameterGroup`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ModifyDBClusterParameterGroup을 참조하세요.

시나리오

Aurora 서버리스 작업 항목 트래커 만들기

다음 코드 예제에서는 Amazon Aurora Serverless 데이터베이스에서 작업 항목을 추적하고 Amazon Simple Email Service(Amazon SES)를 사용하여 보고서를 보내는 웹 애플리케이션을 만드는 방법을 보여줍니다.

SDK for Java 2.x

> Amazon RDS 데이터베이스에 저장된 작업 항목을 추적하고 보고하는 웹 애플리케이션을 만드는 방법을 보여줍니다.

Amazon Aurora Serverless 데이터를 쿼리하고 React 애플리케이션에서 사용하도록 Spring REST API를 설정하는 방법에 대한 지침과 전체 소스 코드는 GitHub에서 전체 예제를 참조하세요.

JDBC API를 사용한 예제를 설정하고 실행하는 방법에 대한 전체 소스 코드와 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Aurora
- Amazon RDS
- Amazon RDS 데이터 서비스
- Amazon SES

Java 2.x용 SDK를 사용하는 Auto Scaling 예제

다음 코드 예제에서는 Auto Scaling과 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시작하기
- 기본 사항
- 작업
- 시나리오

시작하기

Auto Scaling 시작

다음 코드 예제에서는 Auto Scaling 사용을 시작하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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());
        });
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeAutoScalingGroups을 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 시작 템플릿과 가용 영역이 있는 Amazon EC2 Auto Scaling 그룹을 생성하고 실행 중인 인스턴스에 대한 정보를 가져옵니다.
- Amazon CloudWatch 지표 수집 활성화
- 그룹의 원하는 용량을 업데이트하고 인스턴스가 시작될 때까지 기다립니다.
- 그룹에서 인스턴스를 종료합니다.
- 사용자 요청 및 용량 변경에 따라 발생하는 조정 활동을 나열합니다.
- CloudWatch 지표에 대한 통계를 가져온 다음 리소스를 정리합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * 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", "-");
    private static final String ROLES_STACK = "MyCdkAutoScaleStack";
    public static void main(String[] args) throws InterruptedException {
        final String usage = """

                Usage:
                    <groupName>

                Where:
                    groupName - The name of the Auto Scaling group.
                """;

        String groupName = "MyAutoScalingGroup2";
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_WEST_2)
                .build();

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

```
                .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("First, we will create a launch template using a
CloudFormation script");
        CloudFormationHelper.deployCloudFormationStack(ROLES_STACK);
        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputsAsync(ROLES_STACK).join();
        String launchTemplateName = stackOutputs.get("LaunchTemplateNameOutput");
        String vpcZoneId = getVPC(ec2);
        updateTemlate(ec2, launchTemplateName );
        System.out.println("The VPC zone id created by the CloudFormation stack
is"+vpcZoneId);

        System.out.println("1. Create an Auto Scaling group named " + groupName);
        createAutoScalingGroup(autoScalingClient, ec2, 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 and cloud formation
resources");
        CloudFormationHelper.destroyCloudFormationStack(ROLES_STACK);
        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 String getVPC(Ec2Client ec2) {
        try {
            DescribeVpcsRequest request = DescribeVpcsRequest.builder()
                    .filters(f -> f.name("isDefault").values("true"))
                    .build();

            DescribeVpcsResponse response = ec2.describeVpcs(request);

            if (!response.vpcs().isEmpty()) {
                Vpc defaultVpc = response.vpcs().get(0);
                System.out.println("Default VPC ID: " + defaultVpc.vpcId());
                return defaultVpc.vpcId();
            } else {
                System.out.println("No default VPC found.");
                return null; // Return null if no default VPC is found
            }

        } catch (Ec2Exception e) {
            System.err.println("EC2 error: " + e.awsErrorDetails().errorMessage());
            return null; // Return null in case of an error
```

```
        }
    }


    public static void updateTemlate(Ec2Client ec2, String launchTemplateName ) {
        // Step 1: Create new launch template version
        String newAmiId = "ami-0025f0db847eb6254";
        RequestLaunchTemplateData launchTemplateData =
RequestLaunchTemplateData.builder()
                .imageId(newAmiId)
                .build();

        CreateLaunchTemplateVersionRequest createVersionRequest =
CreateLaunchTemplateVersionRequest.builder()
                .launchTemplateName(launchTemplateName)
                .versionDescription("Updated with valid AMI")
                .sourceVersion("1")
                .launchTemplateData(launchTemplateData)
                .build();

        CreateLaunchTemplateVersionResponse createResponse =
ec2.createLaunchTemplateVersion(createVersionRequest);
        int newVersionNumber =
createResponse.launchTemplateVersion().versionNumber().intValue();

        // Step 2: Modify default version
        ModifyLaunchTemplateRequest modifyRequest =
ModifyLaunchTemplateRequest.builder()
                .launchTemplateName(launchTemplateName)
                .defaultVersion(String.valueOf(newVersionNumber))
                .build();

        ec2.modifyLaunchTemplate(modifyRequest);
        System.out.println("Updated launch template to version " + newVersionNumber
+ " with AMI " + newAmiId);
    }


    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,
                                               Ec2Client ec2Client,
                                               String groupName,
                                               String launchTemplateName,
                                               String vpcId) {
        try {
            // Step 1: Get one subnet ID in the given VPC
            DescribeSubnetsRequest subnetRequest = DescribeSubnetsRequest.builder()
```

```
                    .filters(Filter.builder().name("vpc-id").values(vpcId).build())
                    .build();

            DescribeSubnetsResponse subnetResponse =
ec2Client.describeSubnets(subnetRequest);

            if (subnetResponse.subnets().isEmpty()) {
                throw new RuntimeException("No subnets found in VPC: " + vpcId);
            }

            String subnetId = subnetResponse.subnets().get(0).subnetId(); // Use
first subnet
            System.out.println("Using subnet: " + subnetId);

            // Step 2: Create launch template reference
            LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                    .launchTemplateName(launchTemplateName)
                    .build();

            // Step 3: Create Auto Scaling group
            CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .launchTemplate(templateSpecification)
                    .minSize(1)
                    .maxSize(1)
                    .vpcZoneIdentifier(subnetId)  // Correct: subnet ID, not VPC ID
                    .build();

            autoScalingClient.createAutoScalingGroup(request);

            // Step 4: Wait until group is created
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            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 (Ec2Exception | AutoScalingException e) {
            System.err.println("Error: " + 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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하십시오.

- CreateAutoScalingGroup
- DeleteAutoScalingGroup
- DescribeAutoScalingGroups
- DescribeAutoScalingInstances
- DescribeScalingActivities
- DisableMetricsCollection
- EnableMetricsCollection
- SetDesiredCapacity
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

작업

`CreateAutoScalingGroup`

다음 코드 예시는 `CreateAutoScalingGroup`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 자세한 설명은 AWS SDK for Java 2.x API 참조 문서의 CreateAutoScalingGroup을 참조하세요.

DeleteAutoScalingGroup

다음 코드 예시는 DeleteAutoScalingGroup의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 자세한 설명은 AWS SDK for Java 2.x API 참조 문서의 DeleteAutoScalingGroup을 참조하세요.

`DescribeAutoScalingGroups`

다음 코드 예시는 `DescribeAutoScalingGroups`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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 "";
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeAutoScalingGroups를 참조하세요.

`DescribeAutoScalingInstances`

다음 코드 예시는 `DescribeAutoScalingInstances`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeAutoScalingInstances를 참조하세요.

`DescribeScalingActivities`

다음 코드 예시는 `DescribeScalingActivities`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeScalingActivities를 참조하세요.

`DisableMetricsCollection`

다음 코드 예시는 `DisableMetricsCollection`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DisableMetricsCollection을 참조하세요.

`EnableMetricsCollection`

다음 코드 예시는 `EnableMetricsCollection`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 EnableMetricsCollection을 참조하세요.

`SetDesiredCapacity`

다음 코드 예시는 `SetDesiredCapacity`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 SetDesiredCapacity를 참조하세요.

`TerminateInstanceInAutoScalingGroup`

다음 코드 예시는 `TerminateInstanceInAutoScalingGroup`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 TerminateInstanceInAutoScalingGroup을 참조하세요.

`UpdateAutoScalingGroup`

다음 코드 예시는 `UpdateAutoScalingGroup`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateAutoScalingGroup을 참조하세요.

시나리오

복원력이 뛰어난 서비스 구축 및 관리

다음 코드 예제에서는 책, 영화, 노래 추천을 반환하는 로드 밸런싱 웹 서비스를 만드는 방법을 보여줍니다. 이 예제에서는 서비스가 장애에 대응하는 방법과 장애 발생 시 복원력을 높이기 위해 서비스를 재구성하는 방법을 보여줍니다.

- Amazon EC2 Auto Scaling 그룹을 사용하여 시작 템플릿을 기반으로 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스를 생성하고 인스턴스 수를 지정된 범위 내로 유지합니다.
- Elastic Load Balancing으로 HTTP 요청을 처리하고 배포합니다.
- Auto Scaling 그룹의 인스턴스 상태를 모니터링하고 요청을 정상 인스턴스로만 전달합니다.
- 각 EC2 인스턴스에서 Python 웹 서버를 실행하여 HTTP 요청을 처리합니다. 웹 서버는 추천 및 상태 확인으로 응답합니다.
- Amazon DynamoDB 테이블을 사용하여 추천 서비스를 시뮬레이션합니다.
- AWS Systems Manager 파라미터를 업데이트하여 요청 및 상태 확인에 대한 웹 서버 응답을 제어합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
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);
    }
}
```

Auto Scaling과 Amazon EC2 작업을 래핑하는 클래스를 생성합니다.

```
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.");
    }
}
```

ELB 작업을 래핑하는 클래스를 생성합니다.

```
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)
                    .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 "";
    }
}
```

DynamoDB를 사용하여 추천 서비스를 시뮬레이션하는 클래스를 생성합니다.

```
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);
    }
}
```

Systems Manager 작업을 래핑하는 클래스를 생성합니다.

```
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);
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - 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

AWS Batch SDK for Java 2.x를 사용한 예제

다음 코드 예제에서는와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS Batch.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시작하기
- 기본 사항
- 작업

시작하기

안녕하세요 AWS Batch

다음 코드 예제에서는 AWS Batch를 사용하여 시작하는 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
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.batch.BatchAsyncClient;
import software.amazon.awssdk.services.batch.model.JobStatus;
import software.amazon.awssdk.services.batch.model.JobSummary;
import software.amazon.awssdk.services.batch.model.ListJobsRequest;
import software.amazon.awssdk.services.batch.paginators.ListJobsPublisher;
import java.time.Duration;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CompletableFuture;

public class HelloBatch {
    private static BatchAsyncClient batchClient;

    public static void main(String[] args) {
        List<JobSummary> jobs = listJobs("my-job-queue");
        jobs.forEach(job ->
            System.out.printf("Job ID: %s, Job Name: %s, Job Status: %s%n",
                job.jobId(), job.jobName(), job.status())
        );
    }

    public static List<JobSummary> listJobs(String jobQueue) {
        if (jobQueue == null || jobQueue.isEmpty()) {
```

```
            throw new IllegalArgumentException("Job queue cannot be null or empty");
        }

        ListJobsRequest listJobsRequest = ListJobsRequest.builder()
            .jobQueue(jobQueue)
            .jobStatus(JobStatus.SUCCEEDED)
            .build();

        List<JobSummary> jobSummaries = new ArrayList<>();
        ListJobsPublisher listJobsPaginator =
getAsyncClient().listJobsPaginator(listJobsRequest);
        CompletableFuture<Void> future = listJobsPaginator.subscribe(response -> {
            jobSummaries.addAll(response.jobSummaryList());
        });

        future.join();
        return jobSummaries;
    }

    private static BatchAsyncClient getAsyncClient() {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)  // Increase max concurrency to handle more
simultaneous connections.
            .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.
            .retryPolicy(RetryPolicy.builder()  // Add a retry policy to handle
transient errors.
                .numRetries(3)  // Number of retry attempts.
                .build())
            .build();

        if (batchClient == null) {
            batchClient = BatchAsyncClient.builder()
                .region(Region.US_EAST_1)
```

```
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return batchClient;
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 listJobsPaginator를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- AWS Batch 컴퓨팅 환경을 생성합니다.
- 컴퓨팅 환경의 상태를 확인합니다.
- AWS Batch 작업 대기열 및 작업 정의를 설정합니다.
- 작업 정의를 등록합니다.
- AWS Batch 작업을 제출합니다.
- 작업 대기열에 적용할 수 있는 작업 목록을 가져옵니다.
- 작업의 상태를 확인합니다.
- AWS Batch 리소스를 삭제합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

AWS Batch 기능을 보여주는 대화형 시나리오를 실행합니다.

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

```
import software.amazon.awssdk.services.batch.model.BatchException;
import software.amazon.awssdk.services.batch.model.ClientException;
import software.amazon.awssdk.services.batch.model.CreateComputeEnvironmentResponse;
import software.amazon.awssdk.services.batch.model.JobSummary;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeSubnetsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeSubnetsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest;
import software.amazon.awssdk.services.ec2.model.Filter;
import software.amazon.awssdk.services.ec2.model.SecurityGroup;
import software.amazon.awssdk.services.ec2.model.Subnet;
import software.amazon.awssdk.services.ec2.model.Vpc;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * NOTE
 * This scenario submits a job that pulls a Docker image named echo-text from Amazon
 ECR to Amazon Fargate.
 *
 * To place this Docker image on Amazon ECR, run the following Basics scenario.
 *
 * https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/ecr
 *
 */
public class BatchScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    // Define two stacks used in this Basics Scenario.
```

```
    private static final String ROLES_STACK = "RolesStack";
    private static String defaultSubnet;
    private static String defaultSecurityGroup;

    private static final Logger logger =
 LoggerFactory.getLogger(BatchScenario.class);

    public static void main(String[] args) throws InterruptedException {

        BatchActions batchActions = new BatchActions();
        Scanner scanner = new Scanner(System.in);
        String computeEnvironmentName = "my-compute-environment";
        String jobQueueName = "my-job-queue";
        String jobDefinitionName = "my-job-definition";


        // See the NOTE in this Java code example (at start).
        String dockerImage = "dkr.ecr.us-east-1.amazonaws.com/echo-text:echo-text";

        logger.info("""
            AWS Batch is a fully managed batch processing service that dynamically
 provisions the required compute
            resources for batch computing workloads. The Java V2 `BatchAsyncClient`
 allows
            developers to automate the submission, monitoring, and management of
 batch jobs.

            This scenario provides an example of setting up a compute environment,
 job queue and job definition,
            and then submitting a job.

            This scenario submits a job that pulls a Docker image named echo-text
 from Amazon ECR to Amazon Fargate.

            To place this Docker image on Amazon ECR, run the following Basics
 scenario.

            https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/
example_code/ecr

            Let's get started...

            You have two choices:
```

```
            1 - Run the entire program.
            2 - Delete an existing Compute Environment (created from a previous
execution of
            this program that did not complete).
            """);

        while (true) {
            String input = scanner.nextLine();
            if (input.trim().equalsIgnoreCase("1")) {
                logger.info("Continuing with the program...");
               // logger.info("");
                break;
            } else if (input.trim().equalsIgnoreCase("2")) {
                String jobQueueARN = String.valueOf(batchActions.
describeJobQueueAsync(computeEnvironmentName));
                if (!jobQueueARN.isEmpty()) {
                    batchActions.disableJobQueueAsync(jobQueueARN);
                    countdown(1);
                    batchActions.deleteJobQueueAsync(jobQueueARN);
                }

                try {

batchActions.disableComputeEnvironmentAsync(computeEnvironmentName)
                        .exceptionally(ex -> {
                            logger.info("Disable compute environment failed: " +
ex.getMessage());
                            return null;
                        })
                        .join();
                } catch (CompletionException ex) {
                    logger.info("Failed to disable compute environment: " +
ex.getMessage());
                }
                countdown(2);

batchActions.deleteComputeEnvironmentAsync(computeEnvironmentName).join();
                return;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
        System.out.println(DASHES);
```

```
        waitForInputToContinue(scanner);
        // Get an AWS Account id used to retrieve the docker image from Amazon ECR.
        // Create a single-element array to store the `accountId` value.
        String[] accId = new String[1];
        CompletableFuture<String> accountIdFuture = batchActions.getAccountId();
        accountIdFuture.thenAccept(accountId -> {
            logger.info("Account ID: " + accountId);
            accId[0] = accountId;
        }).join();

        dockerImage = accId[0]+"."+dockerImage;

        // Get a default subnet and default security associated with the default
VPC.
        getSubnetSecurityGroup();

        logger.info("Use AWS CloudFormation to create two IAM roles that are
required for this scenario.");
        CloudFormationHelper.deployCloudFormationStack(ROLES_STACK);

        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputs(ROLES_STACK);
        String batchIAMRole = stackOutputs.get("BatchRoleArn");
        String executionRoleARN = stackOutputs.get("EcsRoleArn");

        logger.info("The IAM role needed to interact with AWS Batch is
"+batchIAMRole);
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("1. Create a Batch compute environment");
        logger.info("""
            A compute environment is a resource where you can run your batch jobs.
            After creating a compute environment, you can define job queues and job
definitions to submit jobs for
            execution.

            The benefit of creating a compute environment is it allows you to easily
configure and manage the compute
            resources that will be used to run your Batch jobs. By separating the
compute environment from the job definitions,
            you can easily scale your compute resources up or down as needed,
without having to modify your job definitions.
```

```
            This makes it easier to manage your Batch workloads and ensures that
your jobs have the necessary
            compute resources to run efficiently.
            """);

        waitForInputToContinue(scanner);
        try {
            CompletableFuture<CreateComputeEnvironmentResponse> future =
batchActions.createComputeEnvironmentAsync(computeEnvironmentName, batchIAMRole,
defaultSubnet, defaultSecurityGroup);
            CreateComputeEnvironmentResponse response = future.join();
            logger.info("Compute Environment ARN: " +
response.computeEnvironmentArn());
        } catch (RuntimeException rte) {
            Throwable cause = rte.getCause();
            if (cause instanceof ClientException batchExceptionEx) {
                String myErrorCode =
batchExceptionEx.awsErrorDetails().errorMessage();
                if ("Object already exists".contains(myErrorCode)) {
                    logger.info("The compute environment '" + computeEnvironmentName
+ "' already exists. Moving on...");
                } else {
                    logger.info("Batch error occurred: {} (Code: {})",
batchExceptionEx.getMessage(), batchExceptionEx.awsErrorDetails().errorCode());
                    return;
                }
            } else {
                    logger.info("An unexpected error occurred: {}", (cause != null ?
cause.getMessage() : rte.getMessage()));
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Check the status of the "+computeEnvironmentName +" Compute
Environment.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
batchActions.checkComputeEnvironmentsStatus(computeEnvironmentName);
            String status = future.join();
            logger.info("Compute Environment Status: " + status);
```

```
        } catch (RuntimeException rte) {
            Throwable cause = rte.getCause();
            if (cause instanceof ClientException batchExceptionEx) {
                logger.info("Batch error occurred: {} (Code: {})",
batchExceptionEx.getMessage(), batchExceptionEx.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: " + (cause != null ?
cause.getMessage() : rte.getMessage()));
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Create a job queue");
        logger.info("""
             A job queue is an essential component that helps manage the execution
of your batch jobs.
             It acts as a buffer, where jobs are placed and then scheduled for
execution based on their
             priority and the available resources in the compute environment.
             """);
        waitForInputToContinue(scanner);

        String jobQueueArn = null;
        try {
            CompletableFuture<String> jobQueueFuture =
batchActions.createJobQueueAsync(jobQueueName, computeEnvironmentName);
            jobQueueArn = jobQueueFuture.join();
            logger.info("Job Queue ARN: " + jobQueueArn);

        } catch (RuntimeException rte) {
            Throwable cause = rte.getCause();
            if (cause instanceof BatchException batchExceptionEx) {
                String myErrorCode =
batchExceptionEx.awsErrorDetails().errorMessage();
                if ("Object already exists".contains(myErrorCode)) {
                    logger.info("The job queue '" + jobQueueName + "' already
exists. Moving on...");
                    // Retrieve the ARN of the job queue.
                    CompletableFuture<String> jobQueueArnFuture =
batchActions.getJobQueueARN(jobQueueName);
```

```
                    jobQueueArn = jobQueueArnFuture.join();
                    logger.info("Job Queue ARN: " + jobQueueArn);
                } else {
                    logger.info("Batch error occurred: {} (Code: {})",
batchExceptionEx.getMessage(), batchExceptionEx.awsErrorDetails().errorCode());
                    return;
                }
            } else {
                logger.info("An unexpected error occurred: " + (cause != null ?
cause.getMessage() : rte.getMessage()));
                return; // End the execution
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("4. Register a Job Definition.");
        logger.info("""
            Registering a job in AWS Batch using the Fargate launch type ensures
that all
            necessary parameters, such as the execution role, command to run, and so
on
            are specified and reused across multiple job submissions.

             The job definition pulls a Docker image from Amazon ECR and executes
the Docker image.
            """);

        waitForInputToContinue(scanner);
        String jobARN;
        try {
            String platform = "";
            while (true) {
                logger.info("""
                    On which platform/CPU architecture combination did you build the
Docker image?:
                    1. Windows       X86_64
                    2. Mac or Linux  ARM64
                    3. Mac or Linux  X86_64

                    Please select 1, 2, or 3.
                    """);
                String platAns = scanner.nextLine().trim();
                if (platAns.equals("1")) {
```

```
                    platform = "X86_64";
                    break; // Exit loop since a valid option is selected
                } else if (platAns.equals("2")) {
                    platform = "ARM64";
                    break; // Exit loop since a valid option is selected
                } else if (platAns.equals("3")) {
                    platform = "X86_64";
                    break; // Exit loop since a valid option is selected
                } else {
                    System.out.println("Invalid input. Please select either 1 or
2.");
                }
            }

            jobARN = batchActions.registerJobDefinitionAsync(jobDefinitionName,
executionRoleARN, dockerImage, platform)
                .exceptionally(ex -> {
                    System.err.println("Register job definition failed: " +
ex.getMessage());
                    return null;
                })
                .join();
            if (jobARN != null) {
                logger.info("Job ARN: " + jobARN);
            }
        } catch (RuntimeException rte) {
            logger.error("A Batch exception occurred while registering the job: {}",
rte.getCause() != null ? rte.getCause().getMessage() : rte.getMessage());
            return;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Submit an AWS Batch job from a job definition.");
        waitForInputToContinue(scanner);
        String jobId;
        try {
            jobId = batchActions.submitJobAsync(jobDefinitionName, jobQueueName,
jobARN)
                .exceptionally(ex -> {
                    System.err.println("Submit job failed: " + ex.getMessage());
                    return null;
                })
                .join();
```

```
            logger.info("The job id is "+jobId);
            logger.info("Let's wait 2 minutes for the job to complete");
            countdown(2);

        } catch (RuntimeException rte) {
            logger.error("A Batch exception occurred while submitting the job: {}",
rte.getCause() != null ? rte.getCause().getMessage() : rte.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        logger.info(DASHES);
        logger.info("6. Get a list of jobs applicable to the job queue.");

        waitForInputToContinue(scanner);
        try {
            List<JobSummary> jobs = batchActions.listJobsAsync(jobQueueName);
            jobs.forEach(job ->
                logger.info("Job ID: {}, Job Name: {}, Job Status: {}", job.jobId(),
job.jobName(), job.status()));

        } catch (RuntimeException rte) {
            logger.info("A Batch exception occurred while submitting the job: {}",
rte.getCause() != null ? rte.getCause().getMessage() : rte.getMessage());
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Check the status of job "+jobId);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future = batchActions.describeJobAsync(jobId);
            String jobStatus = future.join();
            logger.info("Job Status: " + jobStatus);

        } catch (RuntimeException rte) {
            logger.info("A Batch exception occurred while submitting the job: {}",
rte.getCause() != null ? rte.getCause().getMessage() : rte.getMessage());
            return;
```

```
        }

        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        logger.info("8. Delete Batch resources");
        logger.info(
            """
            When deleting an AWS Batch compute environment, it does not happen
instantaneously.
            There is typically a delay, similar to some other AWS resources.
            AWS Batch starts the deletion process.
            """);
        logger.info("Would you like to delete the AWS Batch resources such as the
compute environment? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete the AWS ECR resources.");
            logger.info("First, we will deregister the Job Definition.");
            waitForInputToContinue(scanner);
            try {
                batchActions.deregisterJobDefinitionAsync(jobARN)
                    .exceptionally(ex -> {
                        logger.info("Deregister job definition failed: " +
ex.getMessage());
                        return null;
                    })
                    .join();
                logger.info(jobARN + " was deregistered");
            } catch (RuntimeException rte) {
                logger.error("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }

            logger.info("Second, we will disable and then delete the Job Queue.");
            waitForInputToContinue(scanner);
            try {
                batchActions.disableJobQueueAsync(jobQueueArn)
                    .exceptionally(ex -> {
                        logger.info("Disable job queue failed: " + ex.getMessage());
                        return null;
                    })
                    .join();
```

```
                logger.info(jobQueueArn + " was disabled");
            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }

            batchActions.waitForJobQueueToBeDisabledAsync(jobQueueArn);
            try {
                CompletableFuture<Void> future =
batchActions.waitForJobQueueToBeDisabledAsync(jobQueueArn);
                future.join();
                logger.info("Job queue is now disabled.");
            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }

            waitForInputToContinue(scanner);
            try {
                batchActions.deleteJobQueueAsync(jobQueueArn);
                logger.info(jobQueueArn +" was deleted");
            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }
            logger.info("Let's wait 2 minutes for the job queue to be deleted");
            countdown(2);
            waitForInputToContinue(scanner);

            logger.info("Third, we will delete the Compute Environment.");
            waitForInputToContinue(scanner);
            try {
                batchActions.disableComputeEnvironmentAsync(computeEnvironmentName)
                    .exceptionally(ex -> {
                        System.err.println("Disable compute environment failed: " +
ex.getMessage());
                        return null;
                    })
                    .join();
                logger.info("Compute environment disabled") ;
            } catch (RuntimeException rte) {
```

```
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }


batchActions.checkComputeEnvironmentsStatus(computeEnvironmentName).thenAccept(state
-> {
                logger.info("Current State: " + state);
            }).join();

            logger.info("Lets wait 1 min for the compute environment to be
deleted");
            countdown(1);

            try {

batchActions.deleteComputeEnvironmentAsync(computeEnvironmentName).join();
                logger.info(computeEnvironmentName +" was deleted.");

            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }
            waitForInputToContinue(scanner);
            CloudFormationHelper.destroyCloudFormationStack(ROLES_STACK);
        }

        logger.info(DASHES);
        logger.info("This concludes the AWS Batch SDK scenario");
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
```

```
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }

    public static void countdown(int minutes) throws InterruptedException {
        int seconds = 0;
        for (int i = minutes * 60 + seconds; i >= 0; i--) {
            int displayMinutes = i / 60;
            int displaySeconds = i % 60;
            System.out.print(String.format("\r%02d:%02d", displayMinutes,
displaySeconds));
            Thread.sleep(1000); // Wait for 1 second
        }
        logger.info("Countdown complete!");
    }

    private static void getSubnetSecurityGroup() {
        try (Ec2AsyncClient ec2Client = Ec2AsyncClient.create()) {
            CompletableFuture<Vpc> defaultVpcFuture =
ec2Client.describeVpcs(DescribeVpcsRequest.builder()
                            .filters(Filter.builder()
                                    .name("is-default")
                                    .values("true")
                                    .build())
                            .build())
                    .thenApply(response -> response.vpcs().stream()
                            .findFirst()
                            .orElseThrow(() -> new RuntimeException("Default VPC not
found")));

            CompletableFuture<String> defaultSubnetFuture = defaultVpcFuture
                    .thenCompose(vpc ->
ec2Client.describeSubnets(DescribeSubnetsRequest.builder()
                                    .filters(Filter.builder()
                                                    .name("vpc-id")
                                                    .values(vpc.vpcId())
                                                    .build(),
                                            Filter.builder()
                                                    .name("default-for-az")
                                                    .values("true")
                                                    .build())
```

```
                                    .build())
                            .thenApply(DescribeSubnetsResponse::subnets)
                            .thenApply(subnets -> subnets.stream()
                                    .findFirst()
                                    .map(Subnet::subnetId)
                                    .orElseThrow(() -> new RuntimeException("No
 default subnet found"))));

            CompletableFuture<String> defaultSecurityGroupFuture = defaultVpcFuture
                    .thenCompose(vpc ->
 ec2Client.describeSecurityGroups(DescribeSecurityGroupsRequest.builder()
                                    .filters(Filter.builder()
                                                    .name("group-name")
                                                    .values("default")
                                                    .build(),
                                            Filter.builder()
                                                    .name("vpc-id")
                                                    .values(vpc.vpcId())
                                                    .build())
                                    .build())

 .thenApply(DescribeSecurityGroupsResponse::securityGroups)
                            .thenApply(securityGroups -> securityGroups.stream()
                                    .findFirst()
                                    .map(SecurityGroup::groupId)
                                    .orElseThrow(() -> new RuntimeException("No
 default security group found"))));

            defaultSubnet = defaultSubnetFuture.join();
            defaultSecurityGroup = defaultSecurityGroupFuture.join();
        }
    }
}
```

AWS Batch SDK 메서드의 래퍼 클래스입니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
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.batch.BatchAsyncClient;
import software.amazon.awssdk.services.batch.BatchClient;
import software.amazon.awssdk.services.batch.model.AssignPublicIp;
import software.amazon.awssdk.services.batch.model.BatchException;
import software.amazon.awssdk.services.batch.model.CEState;
import software.amazon.awssdk.services.batch.model.CEType;
import software.amazon.awssdk.services.batch.model.CRType;
import software.amazon.awssdk.services.batch.model.ComputeEnvironmentOrder;
import software.amazon.awssdk.services.batch.model.ComputeResource;
import software.amazon.awssdk.services.batch.model.ContainerProperties;
import software.amazon.awssdk.services.batch.model.CreateComputeEnvironmentRequest;
import software.amazon.awssdk.services.batch.model.CreateComputeEnvironmentResponse;
import software.amazon.awssdk.services.batch.model.CreateJobQueueRequest;
import software.amazon.awssdk.services.batch.model.DeleteComputeEnvironmentRequest;
import software.amazon.awssdk.services.batch.model.DeleteComputeEnvironmentResponse;
import software.amazon.awssdk.services.batch.model.DeleteJobQueueRequest;
import software.amazon.awssdk.services.batch.model.DeleteJobQueueResponse;
import software.amazon.awssdk.services.batch.model.DeregisterJobDefinitionRequest;
import software.amazon.awssdk.services.batch.model.DeregisterJobDefinitionResponse;
import
 software.amazon.awssdk.services.batch.model.DescribeComputeEnvironmentsRequest;
import
 software.amazon.awssdk.services.batch.model.DescribeComputeEnvironmentsResponse;
import software.amazon.awssdk.services.batch.model.DescribeJobQueuesRequest;
import software.amazon.awssdk.services.batch.model.DescribeJobQueuesResponse;
import software.amazon.awssdk.services.batch.model.DescribeJobsRequest;
import software.amazon.awssdk.services.batch.model.DescribeJobsResponse;
import software.amazon.awssdk.services.batch.model.JQState;
import software.amazon.awssdk.services.batch.model.JobDefinitionType;
import software.amazon.awssdk.services.batch.model.JobDetail;
import software.amazon.awssdk.services.batch.model.JobQueueDetail;
import software.amazon.awssdk.services.batch.model.JobStatus;
import software.amazon.awssdk.services.batch.model.JobSummary;
import software.amazon.awssdk.services.batch.model.ListJobsRequest;
import software.amazon.awssdk.services.batch.model.RegisterJobDefinitionResponse;
import software.amazon.awssdk.services.batch.model.NetworkConfiguration;
import software.amazon.awssdk.services.batch.model.PlatformCapability;
import software.amazon.awssdk.services.batch.model.RegisterJobDefinitionRequest;
import software.amazon.awssdk.services.batch.model.ResourceRequirement;
import software.amazon.awssdk.services.batch.model.ResourceType;
import software.amazon.awssdk.services.batch.model.RuntimePlatform;
import software.amazon.awssdk.services.batch.model.SubmitJobRequest;
import software.amazon.awssdk.services.batch.model.CreateJobQueueResponse;
```

```
import java.time.Duration;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.atomic.AtomicBoolean;
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.services.batch.model.SubmitJobResponse;
import software.amazon.awssdk.services.batch.model.UpdateComputeEnvironmentRequest;
import software.amazon.awssdk.services.batch.model.UpdateComputeEnvironmentResponse;
import software.amazon.awssdk.services.batch.model.UpdateJobQueueRequest;
import software.amazon.awssdk.services.batch.model.UpdateJobQueueResponse;
import software.amazon.awssdk.services.batch.paginators.ListJobsPublisher;
import software.amazon.awssdk.services.sts.StsAsyncClient;
import software.amazon.awssdk.services.sts.model.GetCallerIdentityResponse;

public class BatchActions {
    private static BatchAsyncClient batchClient;

    private static final Logger logger =
 LoggerFactory.getLogger(BatchActions.class);

    private static BatchAsyncClient getAsyncClient() {
        if (batchClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryPolicy(RetryPolicy.builder()
                    .numRetries(3)
                    .build())
                .build();

            batchClient = BatchAsyncClient.builder()
                .region(Region.US_EAST_1)
```

```
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return batchClient;
    }

    /**
     * Asynchronously creates a new compute environment in AWS Batch.
     *
     * @param computeEnvironmentName the name of the compute environment to create
     * @param batchIAMRole the IAM role to be used by the compute environment
     * @param subnet the subnet ID to be used for the compute environment
     * @param secGroup the security group ID to be used for the compute environment
     * @return a {@link CompletableFuture} representing the asynchronous operation,
which will complete with the
     *         {@link CreateComputeEnvironmentResponse} when the compute environment
has been created
     * @throws BatchException if there is an error creating the compute environment
     * @throws RuntimeException if there is an unexpected error during the operation
     */
    public CompletableFuture<CreateComputeEnvironmentResponse>
createComputeEnvironmentAsync(
        String computeEnvironmentName, String batchIAMRole, String subnet, String
secGroup) {
        CreateComputeEnvironmentRequest environmentRequest =
CreateComputeEnvironmentRequest.builder()
            .computeEnvironmentName(computeEnvironmentName)
            .type(CEType.MANAGED)
            .state(CEState.ENABLED)
            .computeResources(ComputeResource.builder()
                .type(CRType.FARGATE)
                .maxvCpus(256)
                .subnets(Collections.singletonList(subnet))
                .securityGroupIds(Collections.singletonList(secGroup))
                .build())
            .serviceRole(batchIAMRole)
            .build();

        CompletableFuture<CreateComputeEnvironmentResponse> response =
getAsyncClient().createComputeEnvironment(environmentRequest);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
```

```
                String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                throw new RuntimeException(errorMessage, ex);
            }
        });

        return response;
    }

    public CompletableFuture<DeleteComputeEnvironmentResponse>
deleteComputeEnvironmentAsync(String computeEnvironmentName) {
        DeleteComputeEnvironmentRequest deleteComputeEnvironment =
DeleteComputeEnvironmentRequest.builder()
            .computeEnvironment(computeEnvironmentName)
            .build();

        return getAsyncClient().deleteComputeEnvironment(deleteComputeEnvironment)
            .whenComplete((response, ex) -> {
                if (ex != null) {
                    Throwable cause = ex.getCause();
                    if (cause instanceof BatchException) {
                        throw new RuntimeException(cause);
                    } else {
                        throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                    }
                }
            });
    }

    /**
     * Checks the status of the specified compute environment.
     *
     * @param computeEnvironmentName the name of the compute environment to check
     * @return a CompletableFuture containing the status of the compute environment,
or "ERROR" if an exception occurs
     */
    public CompletableFuture<String> checkComputeEnvironmentsStatus(String
computeEnvironmentName) {
        if (computeEnvironmentName == null || computeEnvironmentName.isEmpty()) {
            throw new IllegalArgumentException("Compute environment name cannot be
null or empty");
        }
```

```
        DescribeComputeEnvironmentsRequest environmentsRequest =
DescribeComputeEnvironmentsRequest.builder()
            .computeEnvironments(computeEnvironmentName)
            .build();

        CompletableFuture<DescribeComputeEnvironmentsResponse> response =
getAsyncClient().describeComputeEnvironments(environmentsRequest);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                throw new RuntimeException(errorMessage, ex);
            }
        });

        return response.thenApply(resp -> resp.computeEnvironments().stream()
            .map(env -> env.statusAsString())
            .findFirst()
            .orElse("UNKNOWN"));
    }

    /**
     * Creates a job queue asynchronously.
     *
     * @param jobQueueName the name of the job queue to create
     * @param computeEnvironmentName the name of the compute environment to
associate with the job queue
     * @return a CompletableFuture that completes with the Amazon Resource Name
(ARN) of the job queue
     */
    public CompletableFuture<String> createJobQueueAsync(String jobQueueName, String
computeEnvironmentName) {
        if (jobQueueName == null || jobQueueName.isEmpty()) {
            throw new IllegalArgumentException("Job queue name cannot be null or
empty");
        }
        if (computeEnvironmentName == null || computeEnvironmentName.isEmpty()) {
            throw new IllegalArgumentException("Compute environment name cannot be
null or empty");
        }

        CreateJobQueueRequest request = CreateJobQueueRequest.builder()
            .jobQueueName(jobQueueName)
            .priority(1)
```

```
            .computeEnvironmentOrder(ComputeEnvironmentOrder.builder()
                .computeEnvironment(computeEnvironmentName)
                .order(1)
                .build())
            .build();

        CompletableFuture<CreateJobQueueResponse> response =
getAsyncClient().createJobQueue(request);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                throw new RuntimeException(errorMessage, ex);
            }
        });

        return response.thenApply(CreateJobQueueResponse::jobQueueArn);
    }

    /**
     * Asynchronously lists the jobs in the specified job queue with the given job
status.
     *
     * @param jobQueue the name of the job queue to list jobs from
     * @return a List<JobSummary> that contains the jobs that succeeded
     */
    public List<JobSummary> listJobsAsync(String jobQueue) {
        if (jobQueue == null || jobQueue.isEmpty()) {
            throw new IllegalArgumentException("Job queue cannot be null or empty");
        }

        ListJobsRequest listJobsRequest = ListJobsRequest.builder()
            .jobQueue(jobQueue)
            .jobStatus(JobStatus.SUCCEEDED)  // Filter jobs by status.
            .build();

        List<JobSummary> jobSummaries = new ArrayList<>();
        ListJobsPublisher listJobsPaginator =
getAsyncClient().listJobsPaginator(listJobsRequest);
        CompletableFuture<Void> future = listJobsPaginator.subscribe(response -> {
            jobSummaries.addAll(response.jobSummaryList());
        });
        future.join();
        return jobSummaries;
```

```
    }

    /**
     * Registers a new job definition asynchronously in AWS Batch.
     * <p>
     * When using Fargate as the compute environment, it is crucial to set the
     * {@link NetworkConfiguration} with {@link AssignPublicIp#ENABLED} to
     * ensure proper networking configuration for the Fargate tasks. This
     * allows the tasks to communicate with external services, access the
     * internet, or communicate within a VPC.
     *
     * @param jobDefinitionName the name of the job definition to be registered
     * @param executionRoleARN the ARN (Amazon Resource Name) of the execution role
     *                         that provides permissions for the containers in the
job
     * @param cpuArch a value of either X86_64 or ARM64 required for the service
call
     * @return a CompletableFuture that completes with the ARN of the registered
     *         job definition upon successful execution, or completes exceptionally
with
     *         an error if the registration fails
     */
    public CompletableFuture<String> registerJobDefinitionAsync(String
jobDefinitionName, String executionRoleARN, String image, String cpuArch) {
        NetworkConfiguration networkConfiguration = NetworkConfiguration.builder()
                .assignPublicIp(AssignPublicIp.ENABLED)
                .build();

        ContainerProperties containerProperties = ContainerProperties.builder()
                .image(image)
                .executionRoleArn(executionRoleARN)
                .resourceRequirements(
                        Arrays.asList(
                                ResourceRequirement.builder()
                                        .type(ResourceType.VCPU)
                                        .value("1")
                                        .build(),
                                ResourceRequirement.builder()
                                        .type(ResourceType.MEMORY)
                                        .value("2048")
                                        .build()
                        )
                )
                .networkConfiguration(networkConfiguration)
```

```
                .runtimePlatform(b -> b
                        .cpuArchitecture(cpuArch)
                        .operatingSystemFamily("LINUX"))
                .build();

        RegisterJobDefinitionRequest request =
RegisterJobDefinitionRequest.builder()
                .jobDefinitionName(jobDefinitionName)
                .type(JobDefinitionType.CONTAINER)
                .containerProperties(containerProperties)
                .platformCapabilities(PlatformCapability.FARGATE)
                .build();

        CompletableFuture<String> future = new CompletableFuture<>();
        getAsyncClient().registerJobDefinition(request)
                .thenApply(RegisterJobDefinitionResponse::jobDefinitionArn)
                .whenComplete((result, ex) -> {
                    if (ex != null) {
                        future.completeExceptionally(ex);
                    } else {
                        future.complete(result);
                    }
                });

        return future;
    }

    /**
     * Deregisters a job definition asynchronously.
     *
     * @param jobDefinition the name of the job definition to be deregistered
     * @return a CompletableFuture that completes when the job definition has been
deregistered
     * or an exception has occurred
     */
    public CompletableFuture<DeregisterJobDefinitionResponse>
deregisterJobDefinitionAsync(String jobDefinition) {
        DeregisterJobDefinitionRequest jobDefinitionRequest =
DeregisterJobDefinitionRequest.builder()
            .jobDefinition(jobDefinition)
            .build();

        CompletableFuture<DeregisterJobDefinitionResponse> responseFuture =
getAsyncClient().deregisterJobDefinition(jobDefinitionRequest);
```

```
        responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
            }
        });

        return responseFuture;
    }

    /**
     * Disables the specified job queue asynchronously.
     *
     * @param jobQueueArn the Amazon Resource Name (ARN) of the job queue to be
disabled
     * @return a {@link CompletableFuture} that completes when the job queue update
operation is complete,
     *         or completes exceptionally if an error occurs during the operation
     */
    public CompletableFuture<Void> disableJobQueueAsync(String jobQueueArn) {
        UpdateJobQueueRequest updateRequest = UpdateJobQueueRequest.builder()
            .jobQueue(jobQueueArn)
            .state(JQState.DISABLED)
            .build();

        CompletableFuture<UpdateJobQueueResponse> responseFuture =
getAsyncClient().updateJobQueue(updateRequest);
        return responseFuture.whenComplete((updateResponse, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to update job queue: " +
ex.getMessage(), ex);
            }
        }).thenApply(updateResponse -> null);
    }

    /**
     * Deletes a Batch job queue asynchronously.
     *
     * @param jobQueueArn The Amazon Resource Name (ARN) of the job queue to delete.
     * @return A CompletableFuture that represents the asynchronous deletion of the
job queue.
     *         The future completes when the job queue has been successfully deleted
or if an error occurs.
```

```
    *         If successful, the future will be completed with a {@code Void}
value.
    *         If an error occurs, the future will be completed exceptionally with
the thrown exception.
    */
   public CompletableFuture<Void> deleteJobQueueAsync(String jobQueueArn) {
       DeleteJobQueueRequest deleteRequest = DeleteJobQueueRequest.builder()
           .jobQueue(jobQueueArn)
           .build();

       CompletableFuture<DeleteJobQueueResponse> responseFuture =
getAsyncClient().deleteJobQueue(deleteRequest);
       return responseFuture.whenComplete((deleteResponse, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to delete job queue: " +
ex.getMessage(), ex);
           }
       }).thenApply(deleteResponse -> null);
   }

   /**
    * Asynchronously describes the job queue associated with the specified compute
environment.
    *
    * @param computeEnvironmentName the name of the compute environment to find the
associated job queue for
    * @return a {@link CompletableFuture} that, when completed, contains the job
queue ARN associated with the specified compute environment
    * @throws RuntimeException if the job queue description fails
    */
   public CompletableFuture<String> describeJobQueueAsync(String
computeEnvironmentName) {
       DescribeJobQueuesRequest describeJobQueuesRequest =
DescribeJobQueuesRequest.builder()
           .build();

       CompletableFuture<DescribeJobQueuesResponse> responseFuture =
getAsyncClient().describeJobQueues(describeJobQueuesRequest);
       return responseFuture.whenComplete((describeJobQueuesResponse, ex) -> {
           if (describeJobQueuesResponse != null) {
               String jobQueueARN;
               for (JobQueueDetail jobQueueDetail :
describeJobQueuesResponse.jobQueues()) {
```

```
                    for (ComputeEnvironmentOrder computeEnvironmentOrder :
jobQueueDetail.computeEnvironmentOrder()) {
                        String computeEnvironment =
computeEnvironmentOrder.computeEnvironment();
                        String name = getComputeEnvironmentName(computeEnvironment);
                        if (name.equals(computeEnvironmentName)) {
                            jobQueueARN = jobQueueDetail.jobQueueArn();
                            logger.info("Job queue ARN associated with the compute
environment: " + jobQueueARN);
                        }
                    }
                }
            } else {
                throw new RuntimeException("Failed to describe job queue: " +
ex.getMessage(), ex);
            }
        }).thenApply(describeJobQueuesResponse -> {
            String jobQueueARN = "";
            for (JobQueueDetail jobQueueDetail :
describeJobQueuesResponse.jobQueues()) {
                for (ComputeEnvironmentOrder computeEnvironmentOrder :
jobQueueDetail.computeEnvironmentOrder()) {
                    String computeEnvironment =
computeEnvironmentOrder.computeEnvironment();
                    String name = getComputeEnvironmentName(computeEnvironment);
                    if (name.equals(computeEnvironmentName)) {
                        jobQueueARN = jobQueueDetail.jobQueueArn();
                    }
                }
            }
            return jobQueueARN;
        });
    }

    /**
     * Disables the specified compute environment asynchronously.
     *
     * @param computeEnvironmentName the name of the compute environment to disable
     * @return a CompletableFuture that completes when the compute environment is
disabled
     */
    public CompletableFuture<UpdateComputeEnvironmentResponse>
disableComputeEnvironmentAsync(String computeEnvironmentName) {
```

```
        UpdateComputeEnvironmentRequest updateRequest =
UpdateComputeEnvironmentRequest.builder()
            .computeEnvironment(computeEnvironmentName)
            .state(CEState.DISABLED)
            .build();

        CompletableFuture<UpdateComputeEnvironmentResponse> responseFuture =
getAsyncClient().updateComputeEnvironment(updateRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to disable compute environment: "
+ ex.getMessage(), ex);
            }
        });

        return responseFuture;
    }

    /**
     * Submits a job asynchronously to the AWS Batch service.
     *
     * @param jobDefinitionName the name of the job definition to use
     * @param jobQueueName the name of the job queue to submit the job to
     * @param jobARN the Amazon Resource Name (ARN) of the job definition
     * @return a CompletableFuture that, when completed, contains the job ID of the
submitted job
     */
    public CompletableFuture<String> submitJobAsync(String jobDefinitionName, String
jobQueueName, String jobARN) {
        SubmitJobRequest jobRequest = SubmitJobRequest.builder()
            .jobDefinition(jobARN)
            .jobName(jobDefinitionName)
            .jobQueue(jobQueueName)
            .build();

        CompletableFuture<SubmitJobResponse> responseFuture =
getAsyncClient().submitJob(jobRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
            }
        });
```

```
        return responseFuture.thenApply(SubmitJobResponse::jobId);
    }

    /**
     * Asynchronously retrieves the status of a specific job.
     *
     * @param jobId the ID of the job to retrieve the status for
     * @return a CompletableFuture that completes with the job status
     */
    public CompletableFuture<String> describeJobAsync(String jobId) {
        DescribeJobsRequest describeJobsRequest = DescribeJobsRequest.builder()
            .jobs(jobId)
            .build();

        CompletableFuture<DescribeJobsResponse> responseFuture =
getAsyncClient().describeJobs(describeJobsRequest);
        return responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
            }
        }).thenApply(response -> response.jobs().get(0).status().toString());
    }

    /**
     * Disables the specific job queue using the asynchronous Java client.
     *
     * @param jobQueueArn the Amazon Resource Name (ARN) of the job queue to wait
for
     * @return a {@link CompletableFuture} that completes when the job queue is
disabled
     */
    public CompletableFuture<Void> waitForJobQueueToBeDisabledAsync(String
jobQueueArn) {
        AtomicBoolean isDisabled = new AtomicBoolean(false);
        return CompletableFuture.runAsync(() -> {
            while (!isDisabled.get()) {
                DescribeJobQueuesRequest describeRequest =
DescribeJobQueuesRequest.builder()
                    .jobQueues(jobQueueArn)
                    .build();

                CompletableFuture<DescribeJobQueuesResponse> responseFuture =
getAsyncClient().describeJobQueues(describeRequest);
```

```
                responseFuture.whenComplete((describeResponse, ex) -> {
                    if (describeResponse != null) {
                        for (JobQueueDetail jobQueue : describeResponse.jobQueues())
{
                            if (jobQueue.jobQueueArn().equals(jobQueueArn) &&
jobQueue.state() == JQState.DISABLED) {
                                isDisabled.set(true);
                                break;
                            }
                        }
                    } else {
                        throw new RuntimeException("Error describing job queues",
ex);
                    }
                }).join();

                if (!isDisabled.get()) {
                    try {
                        logger.info("Waiting for job queue to be disabled...");
                        Thread.sleep(5000);
                    } catch (InterruptedException e) {
                        Thread.currentThread().interrupt();
                        throw new RuntimeException("Thread interrupted while waiting
for job queue to be disabled", e);
                    }
                }
            }
        }).whenComplete((result, throwable) -> {
            if (throwable != null) {
                throw new RuntimeException("Error while waiting for job queue to be
disabled", throwable);
            }
        });
    }

    public CompletableFuture<String> getJobQueueARN(String jobQueueName) {
        // Describe the job queue asynchronously
        CompletableFuture<DescribeJobQueuesResponse> describeJobQueuesFuture =
batchClient.describeJobQueues(
            DescribeJobQueuesRequest.builder()
                .jobQueues(jobQueueName)
                .build()
        );
```

```
        // Handle the asynchronous response and return the Job Queue ARN in the
CompletableFuture<String>
        CompletableFuture<String> jobQueueArnFuture = new CompletableFuture<>();
        describeJobQueuesFuture.whenComplete((response, error) -> {
            if (error != null) {
                if (error instanceof BatchException) {
                    logger.info("Batch error: " + ((BatchException)
error).awsErrorDetails().errorMessage());
                } else {
                    logger.info("Error describing job queue: " +
error.getMessage());
                }
                jobQueueArnFuture.completeExceptionally(new RuntimeException("Failed
to retrieve Job Queue ARN", error));
            } else {
                if (response.jobQueues().isEmpty()) {
                    jobQueueArnFuture.completeExceptionally(new
RuntimeException("Job queue not found: " + jobQueueName));
                } else {
                    // Assuming only one job queue is returned for the given name
                    String jobQueueArn = response.jobQueues().get(0).jobQueueArn();
                    jobQueueArnFuture.complete(jobQueueArn);
                }
            }
        });

        return jobQueueArnFuture;
    }

    private static String getComputeEnvironmentName(String computeEnvironment) {
        String[] parts = computeEnvironment.split("/");
        if (parts.length == 2) {
            return parts[1];
        }
        return null;
    }

    public CompletableFuture<String> getAccountId() {
        StsAsyncClient stsAsyncClient = StsAsyncClient.builder()
            .region(Region.US_EAST_1)
            .build();

        return stsAsyncClient.getCallerIdentity()
            .thenApply(GetCallerIdentityResponse::account);
```

```
    }


}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - CreateComputeEnvironment
  - CreateJobQueue
  - DeleteComputeEnvironment
  - DeleteJobQueue
  - DeregisterJobDefinition
  - DescribeComputeEnvironments
  - DescribeJobQueues
  - DescribeJobs
  - ListJobsPaginator
  - RegisterJobDefinition
  - SubmitJob
  - UpdateComputeEnvironment
  - UpdateJobQueue

작업

`CreateComputeEnvironment`

다음 코드 예시는 `CreateComputeEnvironment`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
```

```
    * Asynchronously creates a new compute environment in AWS Batch.
    *
    * @param computeEnvironmentName the name of the compute environment to create
    * @param batchIAMRole the IAM role to be used by the compute environment
    * @param subnet the subnet ID to be used for the compute environment
    * @param secGroup the security group ID to be used for the compute environment
    * @return a {@link CompletableFuture} representing the asynchronous operation,
which will complete with the
    *         {@link CreateComputeEnvironmentResponse} when the compute environment
has been created
    * @throws BatchException if there is an error creating the compute environment
    * @throws RuntimeException if there is an unexpected error during the operation
    */
   public CompletableFuture<CreateComputeEnvironmentResponse>
createComputeEnvironmentAsync(
       String computeEnvironmentName, String batchIAMRole, String subnet, String
secGroup) {
       CreateComputeEnvironmentRequest environmentRequest =
CreateComputeEnvironmentRequest.builder()
           .computeEnvironmentName(computeEnvironmentName)
           .type(CEType.MANAGED)
           .state(CEState.ENABLED)
           .computeResources(ComputeResource.builder()
               .type(CRType.FARGATE)
               .maxvCpus(256)
               .subnets(Collections.singletonList(subnet))
               .securityGroupIds(Collections.singletonList(secGroup))
               .build())
           .serviceRole(batchIAMRole)
           .build();

       CompletableFuture<CreateComputeEnvironmentResponse> response =
getAsyncClient().createComputeEnvironment(environmentRequest);
       response.whenComplete((resp, ex) -> {
           if (ex != null) {
              String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
              throw new RuntimeException(errorMessage, ex);
           }
       });

       return response;
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateComputeEnvironment를 참조하세요.

`CreateJobQueue`

다음 코드 예시는 `CreateJobQueue`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Creates a job queue asynchronously.
    *
    * @param jobQueueName the name of the job queue to create
    * @param computeEnvironmentName the name of the compute environment to
associate with the job queue
    * @return a CompletableFuture that completes with the Amazon Resource Name
(ARN) of the job queue
    */
   public CompletableFuture<String> createJobQueueAsync(String jobQueueName, String
computeEnvironmentName) {
       if (jobQueueName == null || jobQueueName.isEmpty()) {
           throw new IllegalArgumentException("Job queue name cannot be null or
empty");
       }
       if (computeEnvironmentName == null || computeEnvironmentName.isEmpty()) {
           throw new IllegalArgumentException("Compute environment name cannot be
null or empty");
       }

       CreateJobQueueRequest request = CreateJobQueueRequest.builder()
           .jobQueueName(jobQueueName)
           .priority(1)
           .computeEnvironmentOrder(ComputeEnvironmentOrder.builder()
               .computeEnvironment(computeEnvironmentName)
```

```
                .order(1)
                .build())
            .build();

        CompletableFuture<CreateJobQueueResponse> response =
getAsyncClient().createJobQueue(request);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                throw new RuntimeException(errorMessage, ex);
            }
        });

        return response.thenApply(CreateJobQueueResponse::jobQueueArn);
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateJobQueue를 참조하세요.

`DeleteComputeEnvironment`

다음 코드 예시는 `DeleteComputeEnvironment`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public CompletableFuture<DeleteComputeEnvironmentResponse>
deleteComputeEnvironmentAsync(String computeEnvironmentName) {
        DeleteComputeEnvironmentRequest deleteComputeEnvironment =
DeleteComputeEnvironmentRequest.builder()
            .computeEnvironment(computeEnvironmentName)
            .build();

        return getAsyncClient().deleteComputeEnvironment(deleteComputeEnvironment)
            .whenComplete((response, ex) -> {
```

```
                if (ex != null) {
                    Throwable cause = ex.getCause();
                    if (cause instanceof BatchException) {
                        throw new RuntimeException(cause);
                    } else {
                        throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                    }
                }
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteComputeEnvironment를 참조하세요.

`DeleteJobQueue`

다음 코드 예시는 `DeleteJobQueue`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Deletes a Batch job queue asynchronously.
     *
     * @param jobQueueArn The Amazon Resource Name (ARN) of the job queue to delete.
     * @return A CompletableFuture that represents the asynchronous deletion of the
job queue.
     *         The future completes when the job queue has been successfully deleted
or if an error occurs.
     *         If successful, the future will be completed with a {@code Void}
value.
     *         If an error occurs, the future will be completed exceptionally with
the thrown exception.
     */
    public CompletableFuture<Void> deleteJobQueueAsync(String jobQueueArn) {
        DeleteJobQueueRequest deleteRequest = DeleteJobQueueRequest.builder()
```

```
            .jobQueue(jobQueueArn)
            .build();

        CompletableFuture<DeleteJobQueueResponse> responseFuture =
getAsyncClient().deleteJobQueue(deleteRequest);
        return responseFuture.whenComplete((deleteResponse, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete job queue: " +
ex.getMessage(), ex);
            }
        }).thenApply(deleteResponse -> null);
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteJobQueue를 참조하세요.

`DeregisterJobDefinition`

다음 코드 예시는 `DeregisterJobDefinition`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Deregisters a job definition asynchronously.
     *
     * @param jobDefinition the name of the job definition to be deregistered
     * @return a CompletableFuture that completes when the job definition has been
deregistered
     * or an exception has occurred
     */
    public CompletableFuture<DeregisterJobDefinitionResponse>
deregisterJobDefinitionAsync(String jobDefinition) {
        DeregisterJobDefinitionRequest jobDefinitionRequest =
DeregisterJobDefinitionRequest.builder()
            .jobDefinition(jobDefinition)
            .build();
```

```
        CompletableFuture<DeregisterJobDefinitionResponse> responseFuture =
getAsyncClient().deregisterJobDefinition(jobDefinitionRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
            }
        });

        return responseFuture;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeregisterJobDefinition을 참조하세요.

`DescribeComputeEnvironments`

다음 코드 예시는 `DescribeComputeEnvironments`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Checks the status of the specified compute environment.
     *
     * @param computeEnvironmentName the name of the compute environment to check
     * @return a CompletableFuture containing the status of the compute environment,
or "ERROR" if an exception occurs
     */
    public CompletableFuture<String> checkComputeEnvironmentsStatus(String
computeEnvironmentName) {
        if (computeEnvironmentName == null || computeEnvironmentName.isEmpty()) {
            throw new IllegalArgumentException("Compute environment name cannot be
null or empty");
        }
```

```
        DescribeComputeEnvironmentsRequest environmentsRequest =
DescribeComputeEnvironmentsRequest.builder()
            .computeEnvironments(computeEnvironmentName)
            .build();

        CompletableFuture<DescribeComputeEnvironmentsResponse> response =
getAsyncClient().describeComputeEnvironments(environmentsRequest);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                throw new RuntimeException(errorMessage, ex);
            }
        });

        return response.thenApply(resp -> resp.computeEnvironments().stream()
            .map(env -> env.statusAsString())
            .findFirst()
            .orElse("UNKNOWN"));
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeComputeEnvironments를 참조하세요.

`DescribeJobQueues`

다음 코드 예시는 `DescribeJobQueues`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Asynchronously describes the job queue associated with the specified compute
environment.
     *
```

```
    * @param computeEnvironmentName the name of the compute environment to find the
associated job queue for
    * @return a {@link CompletableFuture} that, when completed, contains the job
queue ARN associated with the specified compute environment
    * @throws RuntimeException if the job queue description fails
    */
   public CompletableFuture<String> describeJobQueueAsync(String
computeEnvironmentName) {
       DescribeJobQueuesRequest describeJobQueuesRequest =
DescribeJobQueuesRequest.builder()
           .build();

       CompletableFuture<DescribeJobQueuesResponse> responseFuture =
getAsyncClient().describeJobQueues(describeJobQueuesRequest);
       return responseFuture.whenComplete((describeJobQueuesResponse, ex) -> {
           if (describeJobQueuesResponse != null) {
               String jobQueueARN;
               for (JobQueueDetail jobQueueDetail :
describeJobQueuesResponse.jobQueues()) {
                   for (ComputeEnvironmentOrder computeEnvironmentOrder :
jobQueueDetail.computeEnvironmentOrder()) {
                       String computeEnvironment =
computeEnvironmentOrder.computeEnvironment();
                       String name = getComputeEnvironmentName(computeEnvironment);
                       if (name.equals(computeEnvironmentName)) {
                           jobQueueARN = jobQueueDetail.jobQueueArn();
                           logger.info("Job queue ARN associated with the compute
environment: " + jobQueueARN);
                       }
                   }
               }
           } else {
               throw new RuntimeException("Failed to describe job queue: " +
ex.getMessage(), ex);
           }
       }).thenApply(describeJobQueuesResponse -> {
           String jobQueueARN = "";
           for (JobQueueDetail jobQueueDetail :
describeJobQueuesResponse.jobQueues()) {
               for (ComputeEnvironmentOrder computeEnvironmentOrder :
jobQueueDetail.computeEnvironmentOrder()) {
                   String computeEnvironment =
computeEnvironmentOrder.computeEnvironment();
                   String name = getComputeEnvironmentName(computeEnvironment);
```

```
                    if (name.equals(computeEnvironmentName)) {
                        jobQueueARN = jobQueueDetail.jobQueueArn();
                    }
                }
            }
            return jobQueueARN;
        });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeJobQueues를 참조하세요.

`DescribeJobs`

다음 코드 예시는 `DescribeJobs`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Asynchronously retrieves the status of a specific job.
     *
     * @param jobId the ID of the job to retrieve the status for
     * @return a CompletableFuture that completes with the job status
     */
    public CompletableFuture<String> describeJobAsync(String jobId) {
        DescribeJobsRequest describeJobsRequest = DescribeJobsRequest.builder()
            .jobs(jobId)
            .build();

        CompletableFuture<DescribeJobsResponse> responseFuture =
getAsyncClient().describeJobs(describeJobsRequest);
        return responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
            }
```

```
        }).thenApply(response -> response.jobs().get(0).status().toString());
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeJobs을 참조하세요.

`ListJobsPaginator`

다음 코드 예시는 `ListJobsPaginator`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Asynchronously lists the jobs in the specified job queue with the given job
status.
     *
     * @param jobQueue the name of the job queue to list jobs from
     * @return a List<JobSummary> that contains the jobs that succeeded
     */
    public List<JobSummary> listJobsAsync(String jobQueue) {
        if (jobQueue == null || jobQueue.isEmpty()) {
            throw new IllegalArgumentException("Job queue cannot be null or empty");
        }

        ListJobsRequest listJobsRequest = ListJobsRequest.builder()
            .jobQueue(jobQueue)
            .jobStatus(JobStatus.SUCCEEDED)  // Filter jobs by status.
            .build();

        List<JobSummary> jobSummaries = new ArrayList<>();
        ListJobsPublisher listJobsPaginator =
getAsyncClient().listJobsPaginator(listJobsRequest);
        CompletableFuture<Void> future = listJobsPaginator.subscribe(response -> {
            jobSummaries.addAll(response.jobSummaryList());
        });
        future.join();
```

```
        return jobSummaries;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListJobsPaginator를 참조하세요.

`RegisterJobDefinition`

다음 코드 예시는 `RegisterJobDefinition`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Registers a new job definition asynchronously in AWS Batch.
     * <p>
     * When using Fargate as the compute environment, it is crucial to set the
     * {@link NetworkConfiguration} with {@link AssignPublicIp#ENABLED} to
     * ensure proper networking configuration for the Fargate tasks. This
     * allows the tasks to communicate with external services, access the
     * internet, or communicate within a VPC.
     *
     * @param jobDefinitionName the name of the job definition to be registered
     * @param executionRoleARN the ARN (Amazon Resource Name) of the execution role
     *                         that provides permissions for the containers in the
job
     * @param cpuArch a value of either X86_64 or ARM64 required for the service
call
     * @return a CompletableFuture that completes with the ARN of the registered
     *         job definition upon successful execution, or completes exceptionally
with
     *         an error if the registration fails
     */
    public CompletableFuture<String> registerJobDefinitionAsync(String
jobDefinitionName, String executionRoleARN, String image, String cpuArch) {
        NetworkConfiguration networkConfiguration = NetworkConfiguration.builder()
                .assignPublicIp(AssignPublicIp.ENABLED)
```

```
                .build();

        ContainerProperties containerProperties = ContainerProperties.builder()
                .image(image)
                .executionRoleArn(executionRoleARN)
                .resourceRequirements(
                        Arrays.asList(
                                ResourceRequirement.builder()
                                        .type(ResourceType.VCPU)
                                        .value("1")
                                        .build(),
                                ResourceRequirement.builder()
                                        .type(ResourceType.MEMORY)
                                        .value("2048")
                                        .build()
                        )
                )
                .networkConfiguration(networkConfiguration)
               .runtimePlatform(b -> b
                        .cpuArchitecture(cpuArch)
                        .operatingSystemFamily("LINUX"))
                .build();

        RegisterJobDefinitionRequest request =
RegisterJobDefinitionRequest.builder()
                .jobDefinitionName(jobDefinitionName)
                .type(JobDefinitionType.CONTAINER)
                .containerProperties(containerProperties)
                .platformCapabilities(PlatformCapability.FARGATE)
                .build();

        CompletableFuture<String> future = new CompletableFuture<>();
        getAsyncClient().registerJobDefinition(request)
                .thenApply(RegisterJobDefinitionResponse::jobDefinitionArn)
                .whenComplete((result, ex) -> {
                    if (ex != null) {
                        future.completeExceptionally(ex);
                    } else {
                        future.complete(result);
                    }
                });

        return future;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 RegisterJobDefinition을 참조하세요.

`SubmitJob`

다음 코드 예시는 `SubmitJob`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Submits a job asynchronously to the AWS Batch service.
    *
    * @param jobDefinitionName the name of the job definition to use
    * @param jobQueueName the name of the job queue to submit the job to
    * @param jobARN the Amazon Resource Name (ARN) of the job definition
    * @return a CompletableFuture that, when completed, contains the job ID of the
submitted job
    */
   public CompletableFuture<String> submitJobAsync(String jobDefinitionName, String
jobQueueName, String jobARN) {
       SubmitJobRequest jobRequest = SubmitJobRequest.builder()
           .jobDefinition(jobARN)
           .jobName(jobDefinitionName)
           .jobQueue(jobQueueName)
           .build();

       CompletableFuture<SubmitJobResponse> responseFuture =
getAsyncClient().submitJob(jobRequest);
       responseFuture.whenComplete((response, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
           }
       });
```

```
        return responseFuture.thenApply(SubmitJobResponse::jobId);
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 SubmitJob을 참조하세요.

`UpdateComputeEnvironment`

다음 코드 예시는 `UpdateComputeEnvironment`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Disables the specified compute environment asynchronously.
     *
     * @param computeEnvironmentName the name of the compute environment to disable
     * @return a CompletableFuture that completes when the compute environment is
disabled
     */
    public CompletableFuture<UpdateComputeEnvironmentResponse>
disableComputeEnvironmentAsync(String computeEnvironmentName) {
        UpdateComputeEnvironmentRequest updateRequest =
UpdateComputeEnvironmentRequest.builder()
            .computeEnvironment(computeEnvironmentName)
            .state(CEState.DISABLED)
            .build();

        CompletableFuture<UpdateComputeEnvironmentResponse> responseFuture =
getAsyncClient().updateComputeEnvironment(updateRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to disable compute environment: "
+ ex.getMessage(), ex);
            }
        });
```

```
        return responseFuture;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateComputeEnvironment를 참조하세요.

`UpdateJobQueue`

다음 코드 예시는 `UpdateJobQueue`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Disables the specified job queue asynchronously.
     *
     * @param jobQueueArn the Amazon Resource Name (ARN) of the job queue to be
disabled
     * @return a {@link CompletableFuture} that completes when the job queue update
operation is complete,
     *         or completes exceptionally if an error occurs during the operation
     */
    public CompletableFuture<Void> disableJobQueueAsync(String jobQueueArn) {
        UpdateJobQueueRequest updateRequest = UpdateJobQueueRequest.builder()
            .jobQueue(jobQueueArn)
            .state(JQState.DISABLED)
            .build();

        CompletableFuture<UpdateJobQueueResponse> responseFuture =
getAsyncClient().updateJobQueue(updateRequest);
        return responseFuture.whenComplete((updateResponse, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to update job queue: " +
ex.getMessage(), ex);
            }
        }).thenApply(updateResponse -> null);
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateJobQueue를 참조하세요.

Java 2.x용 SDK를 사용하는 Amazon Bedrock 예제

다음 코드 예제에서는 AWS SDK for Java 2.x Amazon Bedrock에서를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`GetFoundationModel`

다음 코드 예시는 `GetFoundationModel`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

동기식 Amazon Bedrock 클라이언트를 사용하여 파운데이션 모델에 대한 세부 정보를 가져옵니다.

```
/**
 * 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);
       }
   }
```

비동기식 Amazon Bedrock 클라이언트를 사용하여 파운데이션 모델에 대한 세부 정보를 가져옵니다.

```
   /**
    * 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);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetFoundationModel을 참조하세요.

`ListFoundationModels`

다음 코드 예시는 `ListFoundationModels`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

동기식 Amazon Bedrock 클라이언트를 사용하여 사용 가능한 Amazon Bedrock 파운데이션 모델을 나열합니다.

```
    /**
     * 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);
        }
    }
```

비동기식 Amazon Bedrock 클라이언트를 사용하여 사용 가능한 Amazon Bedrock 파운데이션 모델을 나열합니다.

```
    /**
     * 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);
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListFoundationModels를 참조하세요.

Java 2.x용 SDK를 사용하는 Amazon Bedrock 런타임 예제

다음 코드 예제에서는 Amazon Bedrock 런타임과 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시나리오
- Amazon Nova
- Amazon Nova Canvas
- Amazon Titan Image Generator
- Amazon Titan Text Embeddings
- Anthropic Claude
- Cohere Command
- Meta Llama
- Mistral AI
- Stable Diffusion

시나리오

Amazon Bedrock 기반 모델과 상호 작용할 수 있는 플레이그라운드 애플리케이션을 생성

다음 코드 예제는 다양한 양식을 통해 Amazon Bedrock 기반 모델과 상호 작용할 수 있는 플레이그라운드를 생성하는 방법을 보여줍니다.

SDK for Java 2.x

Java 기반 모델(FM) 플레이그라운드는 Amazon Bedrock을 Java와 함께 사용하는 방법을 보여주는 스프링 부트 샘플 애플리케이션입니다. 이 예제는 Java 개발자가 Amazon Bedrock을 사용하여 생성형 AI 지원 애플리케이션을 구축하는 방법을 보여줍니다. 다음 네 가지 플레이그라운드를 사용하여 Amazon Bedrock 기반 모델을 테스트하고 상호 작용할 수 있습니다.

- 텍스트 플레이그라운드.
- 채팅 플레이그라운드.
- 이미지 플레이그라운드.

또한 이 예제에서는 액세스할 수 있는 기본 모델을 해당 특성과 함께 나열하고 표시합니다. 소스 코드와 배포 지침은 GitHub의 프로젝트를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Bedrock 런타임

Amazon Bedrock을 사용하여 텍스트 프롬프트로 동영상 생성

다음 코드 예제에서는 Spring Boot 앱에서 Amazon Bedrock 및 Nova-Reel 모델을 사용하여 텍스트 프롬프트로 동영상을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon Bedrock 및 Nova-Reel을 사용하여 텍스트 프롬프트로 동영상을 생성합니다.

```
import org.springframework.stereotype.Service;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

import java.util.concurrent.CompletableFuture;

@Service
public class VideoGenerationService {

    public GenerateVideoResponse generateVideo(String prompt) {

        // add S3 bucket you want to store your generated videos
        String s3Bucket = "s3://mygeneratedvidoenovatest";


        //Create json request as an instance of Document class
        Document novaRequest = prepareDocument(prompt);

        // Create request
        StartAsyncInvokeRequest request = StartAsyncInvokeRequest.builder()
                .modelId("amazon.nova-reel-v1:0")
                .modelInput(novaRequest)
                .outputDataConfig(AsyncInvokeOutputDataConfig.builder()

 .s3OutputDataConfig(AsyncInvokeS3OutputDataConfig.builder().s3Uri(s3Bucket).build())
```

```
                        .build())
                .build();

        try (BedrockRuntimeAsyncClient bedrockClient =
getBedrockRuntimeAsyncClient()) {
            CompletableFuture<StartAsyncInvokeResponse>
startAsyncInvokeResponseCompletableFuture =
bedrockClient.startAsyncInvoke(request);

            //blocking operation to wait for the AWS API response
            StartAsyncInvokeResponse startAsyncInvokeResponse =
startAsyncInvokeResponseCompletableFuture.get();
            System.out.println("invocation ARN: " +
startAsyncInvokeResponse.invocationArn());

            GenerateVideoResponse response = new GenerateVideoResponse();
            response.setStatus("inProgress");
            response.setExecutionArn(startAsyncInvokeResponse.invocationArn());

            return response;
        } catch (Exception e) {
            System.out.println(e);
            throw new RuntimeException(e);
        }

    }

    public GenerateVideoResponse checkGenerationStatus(String invocationArn) {
        GenerateVideoResponse response = new GenerateVideoResponse();

        try (BedrockRuntimeAsyncClient bedrockClient =
getBedrockRuntimeAsyncClient()) {
            //creating async request to fetch status by invocation Arn
            GetAsyncInvokeRequest asyncRequest =
GetAsyncInvokeRequest.builder().invocationArn(invocationArn).build();

            CompletableFuture<GetAsyncInvokeResponse> asyncInvoke =
bedrockClient.getAsyncInvoke(asyncRequest);

            //blocking operation to wait for the AWS API response
            GetAsyncInvokeResponse asyncInvokeResponse = asyncInvoke.get();
            System.out.println("Invocation status =" +
asyncInvokeResponse.statusAsString());
```

```
            response.setExecutionArn(invocationArn);
            response.setStatus(asyncInvokeResponse.statusAsString());
            return response;
        } catch (Exception e) {
            e.printStackTrace();
            throw new RuntimeException(e);
        }

    }

    private static BedrockRuntimeAsyncClient getBedrockRuntimeAsyncClient() {
        BedrockRuntimeAsyncClient bedrockClient =
 BedrockRuntimeAsyncClient.builder()
                .region(Region.US_EAST_1)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();
        return bedrockClient;
    }

    private static Document prepareDocument(String prompt) {
        Document textToVideoParams = Document.mapBuilder()
                .putString("text", prompt)
                .build();

        Document videoGenerationConfig = Document.mapBuilder()
                .putNumber("durationSeconds", 6)
                .putNumber("fps", 24)
                .putString("dimension", "1280x720")
                .build();

        Document novaRequest = Document.mapBuilder()
                .putString("taskType", "TEXT_VIDEO")
                .putDocument("textToVideoParams", textToVideoParams)
                .putDocument("videoGenerationConfig", videoGenerationConfig)
                .build();
        return novaRequest;
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - GetAsyncInvoke
  - StartAsyncInvoke

Converse API에서 도구 사용

다음 코드 예제에서는 애플리케이션, 생성형 AI 모델, 연결된 도구 또는 API 간에 일반적인 상호 작용을 구축하여 AI와 외부 환경 간의 상호 작용을 매개하는 방법을 보여줍니다. 외부 날씨 API를 AI 모델에 연결하는 예제를 사용하면 사용자 입력에 따라 실시간 날씨 정보를 제공할 수 있습니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

시나리오 흐름의 기본 실행입니다. 이 시나리오는 사용자, Amazon Bedrock Converse API 및 날씨 도구 간의 대화를 오케스트레이션합니다.

```
/*
 This demo illustrates a tool use scenario using Amazon Bedrock's Converse API and a
 weather tool.
 The program interacts with a foundation model on Amazon Bedrock to provide weather
 information based on user
 input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.
 */
public class BedrockScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static String modelId = "amazon.nova-lite-v1:0";
    private static String defaultPrompt = "What is the weather like in Seattle?";
    private static WeatherTool weatherTool = new WeatherTool();

    // The maximum number of recursive calls allowed in the tool use function.
    // This helps prevent infinite loops and potential performance issues.
    private static int maxRecursions = 5;
    static BedrockActions bedrockActions = new BedrockActions();
    public static boolean interactive = true;

    private static final String systemPrompt = """
            You are a weather assistant that provides current weather data for user-
specified locations using only
            the Weather_Tool, which expects latitude and longitude. Infer the
 coordinates from the location yourself.
```

```
            If the user provides coordinates, infer the approximate location and
 refer to it in your response.
            To use the tool, you strictly apply the provided tool specification.

            - Explain your step-by-step process, and give brief updates before each
 step.
            - Only use the Weather_Tool for data. Never guess or make up
 information.
            - Repeat the tool use for subsequent requests if necessary.
            - If the tool errors, apologize, explain weather is unavailable, and
 suggest other options.
            - Report temperatures in °C (°F) and wind in km/h (mph). Keep weather
 reports concise. Sparingly use
              emojis where appropriate.
            - Only respond to weather queries. Remind off-topic users of your
 purpose.
            - Never claim to search online, access external data, or use tools
 besides Weather_Tool.
            - Complete the entire process until you have all required data before
 sending the complete response.
            """;

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("""
                ==================================================
                Welcome to the Amazon Bedrock Tool Use demo!
                ==================================================

                This assistant provides current weather information for user-
specified locations.
                You can ask for weather details by providing the location name or
 coordinates.

                Example queries:
                - What's the weather like in New York?
                - Current weather for latitude 40.70, longitude -74.01
                - Is it warmer in Rome or Barcelona today?

                To exit the program, simply type 'x' and press Enter.

                P.S.: You're not limited to single locations, or even to using
 English!
                Have fun and experiment with the app!
```

```
                """);
        System.out.println(DASHES);

        try {
            runConversation(scanner);

        } catch (Exception ex) {
            System.out.println("There was a problem running the scenario: " +
ex.getMessage());
        }

        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("Amazon Bedrock Converse API with Tool Use Feature
Scenario is complete.");
        System.out.println(DASHES);
    }

    /**
     * Starts the conversation with the user and handles the interaction with
Bedrock.
     */
    private static List<Message> runConversation(Scanner scanner) {
        List<Message> conversation = new ArrayList<>();

        // Get the first user input
        String userInput = getUserInput("Your weather info request:", scanner);
        System.out.println(userInput);

        while (userInput != null) {
            ContentBlock block = ContentBlock.builder()
                    .text(userInput)
                    .build();

            List<ContentBlock> blockList = new ArrayList<>();
            blockList.add(block);

            Message message = Message.builder()
                    .role(ConversationRole.USER)
                    .content(blockList)
                    .build();

            conversation.add(message);
```

```
            // Send the conversation to Amazon Bedrock.
            ConverseResponse bedrockResponse =
sendConversationToBedrock(conversation);

            // Recursively handle the model's response until the model has returned
its final response or the recursion counter has reached 0.
            processModelResponse(bedrockResponse, conversation, maxRecursions);

            // Repeat the loop until the user decides to exit the application.
            userInput = getUserInput("Your weather info request:", scanner);
        }
        printFooter();
        return conversation;
    }

    /**
     * Processes the response from the model and updates the conversation
accordingly.
     *
     * @param modelResponse the response from the model
     * @param conversation  the ongoing conversation
     * @param maxRecursion  the maximum number of recursions allowed
     */
    private static void processModelResponse(ConverseResponse modelResponse,
List<Message> conversation, int maxRecursion) {
        if (maxRecursion <= 0) {
            // Stop the process, the number of recursive calls could indicate an
infinite loop
            System.out.println("\tWarning: Maximum number of recursions reached.
Please try again.");
        }

        // Append the model's response to the ongoing conversation
        conversation.add(modelResponse.output().message());

        String modelResponseVal = modelResponse.stopReasonAsString();
        if (modelResponseVal.compareTo("tool_use") == 0) {
            // If the stop reason is "tool_use", forward everything to the tool use
handler
            handleToolUse(modelResponse.output(), conversation, maxRecursion - 1);
        }

        if (modelResponseVal.compareTo("end_turn") == 0) {
```

```
            // If the stop reason is "end_turn", print the model's response text,
and finish the process

PrintModelResponse(modelResponse.output().message().content().get(0).text());
            if (!interactive) {
                defaultPrompt = "x";
            }
        }
    }

    /**
     * Handles the use of a tool by the model in a conversation.
     *
     * @param modelResponse the response from the model, which may include a tool
use request
     * @param conversation  the current conversation, which will be updated with the
tool use results
     * @param maxRecursion  the maximum number of recursive calls allowed to handle
the model's response
     */
    private static void handleToolUse(ConverseOutput modelResponse, List<Message>
conversation, int maxRecursion) {
        List<ContentBlock> toolResults = new ArrayList<>();

        // The model's response can consist of multiple content blocks
        for (ContentBlock contentBlock : modelResponse.message().content()) {
            if (contentBlock.text() != null && !contentBlock.text().isEmpty()) {
                // If the content block contains text, print it to the console
                PrintModelResponse(contentBlock.text());
            }

            if (contentBlock.toolUse() != null) {
                ToolResponse toolResponse = invokeTool(contentBlock.toolUse());

                // Add the tool use ID and the tool's response to the list of
results
                List<ToolResultContentBlock> contentBlockList = new ArrayList<>();
                ToolResultContentBlock block = ToolResultContentBlock.builder()
                        .json(toolResponse.getContent())
                        .build();
                contentBlockList.add(block);

                ToolResultBlock toolResultBlock = ToolResultBlock.builder()
                        .toolUseId(toolResponse.getToolUseId())
```

```
                        .content(contentBlockList)
                        .build();

                ContentBlock contentBlock1 = ContentBlock.builder()
                        .toolResult(toolResultBlock)
                        .build();

                toolResults.add(contentBlock1);
            }
        }

        // Embed the tool results in a new user message
        Message message = Message.builder()
                .role(ConversationRole.USER)
                .content(toolResults)
                .build();

        // Append the new message to the ongoing conversation
        //conversation.add(message);
        conversation.add(message);

        // Send the conversation to Amazon Bedrock
        var response = sendConversationToBedrock(conversation);

        // Recursively handle the model's response until the model has returned its
final response or the recursion counter has reached 0
        processModelResponse(response, conversation, maxRecursion);
    }

    // Invokes the specified tool with the given payload and returns the tool's
response.
    // If the requested tool does not exist, an error message is returned.
    private static ToolResponse invokeTool(ToolUseBlock payload) {
        String toolName = payload.name();

        if (Objects.equals(toolName, "Weather_Tool")) {
            Map<String, Document> inputData = payload.input().asMap();
            printToolUse(toolName, inputData);

            // Invoke the weather tool with the input data provided
            Document weatherResponse =
weatherTool.fetchWeatherData(inputData.get("latitude").toString(),
inputData.get("longitude").toString());
```

```
            ToolResponse toolResponse = new ToolResponse();
            toolResponse.setContent(weatherResponse);
            toolResponse.setToolUseId(payload.toolUseId());
            return toolResponse;
        } else {
            String errorMessage = "The requested tool with name " + toolName + "
does not exist.";
            System.out.println(errorMessage);
            return null;
        }
    }

    public static void printToolUse(String toolName, Map<String, Document>
inputData) {
        System.out.println("Invoking tool: " + toolName + " with input: " +
inputData.get("latitude").toString() + ", " + inputData.get("longitude").toString()
+ "...");
    }

    private static void PrintModelResponse(String message) {
        System.out.println("\tThe model's response:\n");
        System.out.println(message);
        System.out.println("");
    }

    private static ConverseResponse sendConversationToBedrock(List<Message>
conversation) {
        System.out.println("Calling Bedrock...");

        try {
            return bedrockActions.sendConverseRequestAsync(modelId, systemPrompt,
conversation, weatherTool.getToolSpec());
        } catch (ModelNotReadyException ex) {
             System.err.println("Model is not ready. Please try again later: " +
ex.getMessage());
            throw ex;
        } catch (BedrockRuntimeException ex) {
            System.err.println("Bedrock service error: " + ex.getMessage());
            throw ex;
        } catch (RuntimeException ex) {
            System.err.println("Unexpected error occurred: " + ex.getMessage());
            throw ex;
        }
    }
```

```
   private static ConverseResponse sendConversationToBedrockwithSpec(List<Message>
conversation, ToolSpecification toolSpec) {
       System.out.println("Calling Bedrock...");

       // Send the conversation, system prompt, and tool configuration, and return
the response
       return bedrockActions.sendConverseRequestAsync(modelId, systemPrompt,
conversation, toolSpec);
   }

   public static String getUserInput(String prompt, Scanner scanner) {
       String userInput = defaultPrompt;
       if (interactive) {
           System.out.println("*".repeat(80));
           System.out.println(prompt + " (x to exit): \n\t");
           userInput = scanner.nextLine();
       }

       if (userInput == null || userInput.trim().isEmpty()) {
           return getUserInput("\tPlease enter your weather info request, e.g., the
name of a city", scanner);
       }

       if (userInput.equalsIgnoreCase("x")) {
           return null;
       }

       return userInput;
   }

   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.");
```

```
            }
        }
    }

    public static void printFooter() {
        System.out.println("""
                ================================================
                Thank you for checking out the Amazon Bedrock Tool Use demo. We hope
 you
                learned something new, or got some inspiration for your own apps
 today!

                For more Bedrock examples in different programming languages, have a
 look at:
                https://docs.aws.amazon.com/bedrock/latest/userguide/
service_code_examples.html
                ================================================
                """);
    }
}
```

데모에서 사용하는 날씨 도구입니다. 이 파일은 도구 사양을 정의하고 Open-Meteo API를 사용하여 날씨 데이터를 검색하는 로직을 구현합니다.

```
public class WeatherTool {

    private static final Logger logger = LoggerFactory.getLogger(WeatherTool.class);
    private static java.net.http.HttpClient httpClient = null;

    /**
     * Returns the JSON Schema specification for the Weather tool. The tool
 specification
     * defines the input schema and describes the tool's functionality.
     * For more information, see https://json-schema.org/understanding-json-schema/
reference.
     *
     * @return The tool specification for the Weather tool.
     */
    public ToolSpecification getToolSpec() {
        Map<String, Document> latitudeMap = new HashMap<>();
        latitudeMap.put("type", Document.fromString("string"));
```

```
        latitudeMap.put("description", Document.fromString("Geographical WGS84
latitude of the location."));

        // Create the nested "longitude" object
        Map<String, Document> longitudeMap = new HashMap<>();
        longitudeMap.put("type", Document.fromString("string"));
        longitudeMap.put("description", Document.fromString("Geographical WGS84
longitude of the location."));

        // Create the "properties" object
        Map<String, Document> propertiesMap = new HashMap<>();
        propertiesMap.put("latitude", Document.fromMap(latitudeMap));
        propertiesMap.put("longitude", Document.fromMap(longitudeMap));

        // Create the "required" array
        List<Document> requiredList = new ArrayList<>();
        requiredList.add(Document.fromString("latitude"));
        requiredList.add(Document.fromString("longitude"));

        // Create the root object
        Map<String, Document> rootMap = new HashMap<>();
        rootMap.put("type", Document.fromString("object"));
        rootMap.put("properties", Document.fromMap(propertiesMap));
        rootMap.put("required", Document.fromList(requiredList));

        // Now create the Document representing the JSON schema
        Document document = Document.fromMap(rootMap);

        ToolSpecification specification = ToolSpecification.builder()
            .name("Weather_Tool")
            .description("Get the current weather for a given location, based on its
WGS84 coordinates.")
            .inputSchema(ToolInputSchema.builder()
                .json(document)
                .build())
            .build();

        return specification;
    }

    /**
     * Fetches weather data for the given latitude and longitude.
     *
     * @param latitude  the latitude coordinate
```

```
     * @param longitude the longitude coordinate
     * @return a {@link CompletableFuture} containing the weather data as a JSON
 string
     */
    public Document fetchWeatherData(String latitude, String longitude) {
        HttpClient httpClient = HttpClient.newHttpClient();

        // Ensure no extra double quotes
        latitude = latitude.replace("\"", "");
        longitude = longitude.replace("\"", "");

        String endpoint = "https://api.open-meteo.com/v1/forecast";
        String url = String.format("%s?latitude=%s&longitude=
%s&current_weather=True", endpoint, latitude, longitude);

        HttpRequest request = HttpRequest.newBuilder()
            .uri(URI.create(url))
            .build();

        try {
            HttpResponse<String> response = httpClient.send(request,
 HttpResponse.BodyHandlers.ofString());
            if (response.statusCode() == 200) {
                String weatherJson = response.body();
                System.out.println(weatherJson);
                ObjectMapper objectMapper = new ObjectMapper();
                Map<String, Object> rawMap = objectMapper.readValue(weatherJson, new
 TypeReference<Map<String, Object>>() {});
                Map<String, Document> documentMap = convertToDocumentMap(rawMap);


                Document weatherDocument = Document.fromMap(documentMap);
                System.out.println(weatherDocument);
                return weatherDocument;
            } else {
                throw new RuntimeException("Error fetching weather data: " +
 response.statusCode());
            }
        } catch (Exception e) {
            System.out.println("Error fetching weather data: " + e.getMessage());
            throw new RuntimeException("Error fetching weather data", e);
        }

    }
```

```
    private static Map<String, Document> convertToDocumentMap(Map<String, Object>
 inputMap) {
        Map<String, Document> result = new HashMap<>();
        for (Map.Entry<String, Object> entry : inputMap.entrySet()) {
            result.put(entry.getKey(), convertToDocument(entry.getValue()));
        }
        return result;
    }

    // Convert different types of Objects to Document
    private static Document convertToDocument(Object value) {
        if (value instanceof Map) {
            return Document.fromMap(convertToDocumentMap((Map<String, Object>)
 value));
        } else if (value instanceof Integer) {
            return Document.fromNumber(SdkNumber.fromInteger((Integer) value));
        } else if (value instanceof Double) {  //
            return Document.fromNumber(SdkNumber.fromDouble((Double) value));
        } else if (value instanceof Boolean) {
            return Document.fromBoolean((Boolean) value);
        } else if (value instanceof String) {
            return Document.fromString((String) value);
        }
        return Document.fromNull(); // Handle null values safely
    }
}
```

도구 구성이 포함된 Converse API 작업입니다.

```
    /**
     * Sends an asynchronous converse request to the AI model.
     *
     * @param modelId      the unique identifier of the AI model to be used for the
 converse request
     * @param systemPrompt the system prompt to be included in the converse request
     * @param conversation a list of messages representing the conversation history
     * @param toolSpec     the specification of the tool to be used in the converse
 request
     * @return the converse response received from the AI model
     */
```

```
    public ConverseResponse sendConverseRequestAsync(String modelId, String
systemPrompt, List<Message> conversation, ToolSpecification toolSpec) {
        List<Tool> toolList = new ArrayList<>();
        Tool tool = Tool.builder()
            .toolSpec(toolSpec)
            .build();

        toolList.add(tool);

        ToolConfiguration configuration = ToolConfiguration.builder()
            .tools(toolList)
            .build();

        SystemContentBlock block = SystemContentBlock.builder()
            .text(systemPrompt)
            .build();

        ConverseRequest request = ConverseRequest.builder()
            .modelId(modelId)
            .system(block)
            .messages(conversation)
            .toolConfig(configuration)
            .build();

        try {
            ConverseResponse response = getClient().converse(request).join();
            return response;

        } catch (ModelNotReadyException ex) {
            throw new RuntimeException("Model is not ready: " + ex.getMessage(),
ex);
        } catch (BedrockRuntimeException ex) {
            throw new RuntimeException("Failed to converse with Bedrock model: " +
ex.getMessage(), ex);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Converse를 참조하세요.

Amazon Nova

Converse

다음 코드 예제에서는 Bedrock의 Converse API를 사용하여 Amazon Nova로 텍스트 메시지를 전송하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Bedrock의 Converse API를 비동기 Java 클라이언트와 함께 사용하여 Amazon Nova로 텍스트 메시지를 전송합니다.

```
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.*;

import java.util.concurrent.CompletableFuture;

/**
 * This example demonstrates how to use the Amazon Nova foundation models
 * with an asynchronous Amazon Bedrock runtime client to generate text.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure and send a request
 * - Process the response
 */
public class ConverseAsync {

    public static String converseAsync() {

        // Step 1: Create the Amazon Bedrock runtime client
        // The runtime client handles the communication with AI models on Amazon
 Bedrock
```

```
        BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Step 2: Specify which model to use
        // Available Amazon Nova models and their characteristics:
        // - Amazon Nova Micro: Text-only model optimized for lowest latency and
 cost
        // - Amazon Nova Lite:  Fast, low-cost multimodal model for image, video,
 and text
        // - Amazon Nova Pro:   Advanced multimodal model balancing accuracy, speed,
 and cost
        //
        // For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        String modelId = "amazon.nova-lite-v1:0";

        // Step 3: Create the message
        // The message includes the text prompt and specifies that it comes from the
 user
        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();

        // Step 4: Configure the request
        // Optional parameters to control the model's response:
        // - maxTokens: maximum number of tokens to generate
        // - temperature: randomness (max: 1.0, default: 0.7)
        //   OR
        // - topP: diversity of word choice (max: 1.0, default: 0.9)
        // Note: Use either temperature OR topP, but not both
        ConverseRequest request = ConverseRequest.builder()
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                                .maxTokens(500)     // The maximum response length
                                .temperature(0.5F)  // Using temperature for
 randomness control
```

```
                        //.topP(0.9F)       // Alternative: use topP instead of
 temperature
                ).build();

        // Step 5: Send and process the request asynchronously
        // - Send the request to the model
        // - Extract and return the generated text from the response
        try {
            CompletableFuture<ConverseResponse> asyncResponse =
 client.converse(request);
            return asyncResponse.thenApply(
                    response -> response.output().message().content().get(0).text()
            ).get();

        } catch (Exception e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        String response = converseAsync();
        System.out.println(response);
    }
}
```

Bedrock의 Converse API를 사용하여 Amazon Nova로 텍스트 메시지를 전송합니다.

```
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.*;

/**
 * This example demonstrates how to use the Amazon Nova foundation models
 * with a synchronous Amazon Bedrock runtime client to generate text.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
```

```
 * - Configure and send a request
 * - Process the response
 */
public class Converse {

    public static String converse() {

        // Step 1: Create the Amazon Bedrock runtime client
        // The runtime client handles the communication with AI models on Amazon
 Bedrock
        BedrockRuntimeClient client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Step 2: Specify which model to use
        // Available Amazon Nova models and their characteristics:
        // - Amazon Nova Micro: Text-only model optimized for lowest latency and
 cost
        // - Amazon Nova Lite:  Fast, low-cost multimodal model for image, video,
 and text
        // - Amazon Nova Pro:   Advanced multimodal model balancing accuracy, speed,
 and cost
        //
        // For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        String modelId = "amazon.nova-lite-v1:0";

        // Step 3: Create the message
        // The message includes the text prompt and specifies that it comes from the
 user
        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();

        // Step 4: Configure the request
        // Optional parameters to control the model's response:
        // - maxTokens: maximum number of tokens to generate
        // - temperature: randomness (max: 1.0, default: 0.7)
        //   OR
```

```
        // - topP: diversity of word choice (max: 1.0, default: 0.9)
        // Note: Use either temperature OR topP, but not both
        ConverseRequest request = ConverseRequest.builder()
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                                .maxTokens(500)     // The maximum response length
                                .temperature(0.5F)  // Using temperature for
 randomness control
                        //.topP(0.9F)       // Alternative: use topP instead of
 temperature
                ).build();

        // Step 5: Send and process the request
        // - Send the request to the model
        // - Extract and return the generated text from the response
        try {
            ConverseResponse response = client.converse(request);
            return response.output().message().content().get(0).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) {
        String response = converse();
        System.out.println(response);
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Converse를 참조하세요.

ConverseStream

다음 코드 예제에서는 Bedrock의 Converse API를 사용하여 Amazon Nova에 텍스트 메시지를 전송하고 응답 스트림을 실시간으로 처리하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Bedrock의 Converse API를 사용하여 Amazon Nova로 텍스트 메시지를 전송하고 응답 스트림을 실시간으로 처리합니다.

```
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.*;

import java.util.concurrent.ExecutionException;

/**
 * This example demonstrates how to use the Amazon Nova foundation models with an
 * asynchronous Amazon Bedrock runtime client to generate streaming text responses.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure a streaming request
 * - Set up a stream handler to process the response chunks
 * - Process the streaming response
 */
public class ConverseStream {

    public static void converseStream() {

        // Step 1: Create the Amazon Bedrock runtime client
        // The runtime client handles the communication with AI models on Amazon
 Bedrock
        BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Step 2: Specify which model to use
```

```
        // Available Amazon Nova models and their characteristics:
        // - Amazon Nova Micro: Text-only model optimized for lowest latency and
 cost
        // - Amazon Nova Lite:  Fast, low-cost multimodal model for image, video,
 and text
        // - Amazon Nova Pro:   Advanced multimodal model balancing accuracy, speed,
 and cost
        //
        // For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        String modelId = "amazon.nova-lite-v1:0";

        // Step 3: Create the message
        // The message includes the text prompt and specifies that it comes from the
 user
        var inputText = "Describe the purpose of a 'hello world' program in one
 paragraph";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Step 4: Configure the request
        // Optional parameters to control the model's response:
        // - maxTokens: maximum number of tokens to generate
        // - temperature: randomness (max: 1.0, default: 0.7)
        //   OR
        // - topP: diversity of word choice (max: 1.0, default: 0.9)
        // Note: Use either temperature OR topP, but not both
        ConverseStreamRequest request = ConverseStreamRequest.builder()
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                                .maxTokens(500)     // The maximum response length
                                .temperature(0.5F)  // Using temperature for
 randomness control
                        //.topP(0.9F)       // Alternative: use topP instead of
 temperature
                ).build();

        // Step 5: Set up the stream handler
        // The stream handler processes chunks of the response as they arrive
        // - onContentBlockDelta: Processes each text chunk
```

```
        // - onError: Handles any errors during streaming
        var streamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            System.out.print(chunk.delta().text());
                            System.out.flush();  // Ensure immediate output of each
 chunk
                        }).build())
                .onError(err -> System.err.printf("Can't invoke '%s': %s", modelId,
 err.getMessage()))
                .build();

        // Step 6: Send the streaming request and process the response
        // - Send the request to the model
        // - Attach the handler to process response chunks as they arrive
        // - Handle any errors during streaming
        try {
            client.converseStream(request, streamHandler).get();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
        }
    }

    public static void main(String[] args) {
        converseStream();
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ConverseStream을 참조하세요.

시나리오: Converse API에서 도구 사용

다음 코드 예제에서는 애플리케이션, 생성형 AI 모델, 연결된 도구 또는 API 간에 일반적인 상호 작용을 구축하여 AI와 외부 환경 간의 상호 작용을 매개하는 방법을 보여줍니다. 외부 날씨 API를 AI 모델에 연결하는 예제를 사용하면 사용자 입력에 따라 실시간 날씨 정보를 제공할 수 있습니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

시나리오 흐름의 기본 실행입니다. 이 시나리오는 사용자, Amazon Bedrock Converse API 및 날씨 도구 간의 대화를 오케스트레이션합니다.

```
/*
 This demo illustrates a tool use scenario using Amazon Bedrock's Converse API and a
 weather tool.
 The program interacts with a foundation model on Amazon Bedrock to provide weather
 information based on user
 input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.
 */
public class BedrockScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static String modelId = "amazon.nova-lite-v1:0";
    private static String defaultPrompt = "What is the weather like in Seattle?";
    private static WeatherTool weatherTool = new WeatherTool();

    // The maximum number of recursive calls allowed in the tool use function.
    // This helps prevent infinite loops and potential performance issues.
    private static int maxRecursions = 5;
    static BedrockActions bedrockActions = new BedrockActions();
    public static boolean interactive = true;

    private static final String systemPrompt = """
            You are a weather assistant that provides current weather data for user-
specified locations using only
            the Weather_Tool, which expects latitude and longitude. Infer the
 coordinates from the location yourself.
            If the user provides coordinates, infer the approximate location and
 refer to it in your response.
            To use the tool, you strictly apply the provided tool specification.

            - Explain your step-by-step process, and give brief updates before each
 step.
```

```
            - Only use the Weather_Tool for data. Never guess or make up
 information.
            - Repeat the tool use for subsequent requests if necessary.
            - If the tool errors, apologize, explain weather is unavailable, and
 suggest other options.
            - Report temperatures in °C (°F) and wind in km/h (mph). Keep weather
 reports concise. Sparingly use
              emojis where appropriate.
            - Only respond to weather queries. Remind off-topic users of your
 purpose.
            - Never claim to search online, access external data, or use tools
 besides Weather_Tool.
            - Complete the entire process until you have all required data before
 sending the complete response.
            """;

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("""
                ==================================================
                Welcome to the Amazon Bedrock Tool Use demo!
                ==================================================

                This assistant provides current weather information for user-
specified locations.
                You can ask for weather details by providing the location name or
 coordinates.

                Example queries:
                - What's the weather like in New York?
                - Current weather for latitude 40.70, longitude -74.01
                - Is it warmer in Rome or Barcelona today?

                To exit the program, simply type 'x' and press Enter.

                P.S.: You're not limited to single locations, or even to using
 English!
                Have fun and experiment with the app!
                """);
        System.out.println(DASHES);

        try {
            runConversation(scanner);
```

```
        } catch (Exception ex) {
            System.out.println("There was a problem running the scenario: " +
ex.getMessage());
        }

        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("Amazon Bedrock Converse API with Tool Use Feature
Scenario is complete.");
        System.out.println(DASHES);
    }

    /**
     * Starts the conversation with the user and handles the interaction with
Bedrock.
     */
    private static List<Message> runConversation(Scanner scanner) {
        List<Message> conversation = new ArrayList<>();

        // Get the first user input
        String userInput = getUserInput("Your weather info request:", scanner);
        System.out.println(userInput);

        while (userInput != null) {
            ContentBlock block = ContentBlock.builder()
                    .text(userInput)
                    .build();

            List<ContentBlock> blockList = new ArrayList<>();
            blockList.add(block);

            Message message = Message.builder()
                    .role(ConversationRole.USER)
                    .content(blockList)
                    .build();

            conversation.add(message);

            // Send the conversation to Amazon Bedrock.
            ConverseResponse bedrockResponse =
sendConversationToBedrock(conversation);
```

```
            // Recursively handle the model's response until the model has returned
its final response or the recursion counter has reached 0.
            processModelResponse(bedrockResponse, conversation, maxRecursions);

            // Repeat the loop until the user decides to exit the application.
            userInput = getUserInput("Your weather info request:", scanner);
        }
        printFooter();
        return conversation;
    }

    /**
     * Processes the response from the model and updates the conversation
accordingly.
     *
     * @param modelResponse the response from the model
     * @param conversation  the ongoing conversation
     * @param maxRecursion  the maximum number of recursions allowed
     */
    private static void processModelResponse(ConverseResponse modelResponse,
List<Message> conversation, int maxRecursion) {
        if (maxRecursion <= 0) {
            // Stop the process, the number of recursive calls could indicate an
infinite loop
            System.out.println("\tWarning: Maximum number of recursions reached.
Please try again.");
        }

        // Append the model's response to the ongoing conversation
        conversation.add(modelResponse.output().message());

        String modelResponseVal = modelResponse.stopReasonAsString();
        if (modelResponseVal.compareTo("tool_use") == 0) {
            // If the stop reason is "tool_use", forward everything to the tool use
handler
            handleToolUse(modelResponse.output(), conversation, maxRecursion - 1);
        }

        if (modelResponseVal.compareTo("end_turn") == 0) {
            // If the stop reason is "end_turn", print the model's response text,
and finish the process

PrintModelResponse(modelResponse.output().message().content().get(0).text());
            if (!interactive) {
```

```
                defaultPrompt = "x";
            }
        }
    }

    /**
     * Handles the use of a tool by the model in a conversation.
     *
     * @param modelResponse the response from the model, which may include a tool
use request
     * @param conversation  the current conversation, which will be updated with the
tool use results
     * @param maxRecursion  the maximum number of recursive calls allowed to handle
the model's response
     */
    private static void handleToolUse(ConverseOutput modelResponse, List<Message>
conversation, int maxRecursion) {
        List<ContentBlock> toolResults = new ArrayList<>();

        // The model's response can consist of multiple content blocks
        for (ContentBlock contentBlock : modelResponse.message().content()) {
            if (contentBlock.text() != null && !contentBlock.text().isEmpty()) {
                // If the content block contains text, print it to the console
                PrintModelResponse(contentBlock.text());
            }

            if (contentBlock.toolUse() != null) {
                ToolResponse toolResponse = invokeTool(contentBlock.toolUse());

                // Add the tool use ID and the tool's response to the list of
results
                List<ToolResultContentBlock> contentBlockList = new ArrayList<>();
                ToolResultContentBlock block = ToolResultContentBlock.builder()
                        .json(toolResponse.getContent())
                        .build();
                contentBlockList.add(block);

                ToolResultBlock toolResultBlock = ToolResultBlock.builder()
                        .toolUseId(toolResponse.getToolUseId())
                        .content(contentBlockList)
                        .build();

                ContentBlock contentBlock1 = ContentBlock.builder()
                        .toolResult(toolResultBlock)
```

```
                        .build();

                toolResults.add(contentBlock1);
            }
        }

        // Embed the tool results in a new user message
        Message message = Message.builder()
                .role(ConversationRole.USER)
                .content(toolResults)
                .build();

        // Append the new message to the ongoing conversation
        //conversation.add(message);
        conversation.add(message);

        // Send the conversation to Amazon Bedrock
        var response = sendConversationToBedrock(conversation);

        // Recursively handle the model's response until the model has returned its
final response or the recursion counter has reached 0
        processModelResponse(response, conversation, maxRecursion);
    }

    // Invokes the specified tool with the given payload and returns the tool's
response.
    // If the requested tool does not exist, an error message is returned.
    private static ToolResponse invokeTool(ToolUseBlock payload) {
        String toolName = payload.name();

        if (Objects.equals(toolName, "Weather_Tool")) {
            Map<String, Document> inputData = payload.input().asMap();
            printToolUse(toolName, inputData);

            // Invoke the weather tool with the input data provided
            Document weatherResponse =
weatherTool.fetchWeatherData(inputData.get("latitude").toString(),
inputData.get("longitude").toString());

            ToolResponse toolResponse = new ToolResponse();
            toolResponse.setContent(weatherResponse);
            toolResponse.setToolUseId(payload.toolUseId());
            return toolResponse;
        } else {
```

```
            String errorMessage = "The requested tool with name " + toolName + "
does not exist.";
            System.out.println(errorMessage);
            return null;
        }
    }

    public static void printToolUse(String toolName, Map<String, Document>
inputData) {
        System.out.println("Invoking tool: " + toolName + " with input: " +
inputData.get("latitude").toString() + ", " + inputData.get("longitude").toString()
+ "...");
    }

    private static void PrintModelResponse(String message) {
        System.out.println("\tThe model's response:\n");
        System.out.println(message);
        System.out.println("");
    }

    private static ConverseResponse sendConversationToBedrock(List<Message>
conversation) {
        System.out.println("Calling Bedrock...");

        try {
            return bedrockActions.sendConverseRequestAsync(modelId, systemPrompt,
conversation, weatherTool.getToolSpec());
        } catch (ModelNotReadyException ex) {
             System.err.println("Model is not ready. Please try again later: " +
ex.getMessage());
            throw ex;
        } catch (BedrockRuntimeException ex) {
            System.err.println("Bedrock service error: " + ex.getMessage());
            throw ex;
        } catch (RuntimeException ex) {
            System.err.println("Unexpected error occurred: " + ex.getMessage());
            throw ex;
        }
    }

    private static ConverseResponse sendConversationToBedrockwithSpec(List<Message>
conversation, ToolSpecification toolSpec) {
        System.out.println("Calling Bedrock...");
```

```
        // Send the conversation, system prompt, and tool configuration, and return
the response
        return bedrockActions.sendConverseRequestAsync(modelId, systemPrompt,
conversation, toolSpec);
    }

    public static String getUserInput(String prompt, Scanner scanner) {
        String userInput = defaultPrompt;
        if (interactive) {
            System.out.println("*".repeat(80));
            System.out.println(prompt + " (x to exit): \n\t");
            userInput = scanner.nextLine();
        }

        if (userInput == null || userInput.trim().isEmpty()) {
            return getUserInput("\tPlease enter your weather info request, e.g., the
name of a city", scanner);
        }

        if (userInput.equalsIgnoreCase("x")) {
            return null;
        }

        return userInput;
    }

    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.");
            }
        }
    }

    public static void printFooter() {
```

```
        System.out.println("""
                ================================================
                Thank you for checking out the Amazon Bedrock Tool Use demo. We hope
 you
                learned something new, or got some inspiration for your own apps
 today!

                For more Bedrock examples in different programming languages, have a
 look at:
                https://docs.aws.amazon.com/bedrock/latest/userguide/
service_code_examples.html
                ================================================
                """);
    }
}
```

데모에서 사용하는 날씨 도구입니다. 이 파일은 도구 사양을 정의하고 Open-Meteo API를 사용하여 날씨 데이터를 검색하는 로직을 구현합니다.

```
public class WeatherTool {

    private static final Logger logger = LoggerFactory.getLogger(WeatherTool.class);
    private static java.net.http.HttpClient httpClient = null;

    /**
     * Returns the JSON Schema specification for the Weather tool. The tool
 specification
     * defines the input schema and describes the tool's functionality.
     * For more information, see https://json-schema.org/understanding-json-schema/
reference.
     *
     * @return The tool specification for the Weather tool.
     */
    public ToolSpecification getToolSpec() {
        Map<String, Document> latitudeMap = new HashMap<>();
        latitudeMap.put("type", Document.fromString("string"));
        latitudeMap.put("description", Document.fromString("Geographical WGS84
 latitude of the location."));

        // Create the nested "longitude" object
        Map<String, Document> longitudeMap = new HashMap<>();
        longitudeMap.put("type", Document.fromString("string"));
```

```
        longitudeMap.put("description", Document.fromString("Geographical WGS84
longitude of the location."));

        // Create the "properties" object
        Map<String, Document> propertiesMap = new HashMap<>();
        propertiesMap.put("latitude", Document.fromMap(latitudeMap));
        propertiesMap.put("longitude", Document.fromMap(longitudeMap));

        // Create the "required" array
        List<Document> requiredList = new ArrayList<>();
        requiredList.add(Document.fromString("latitude"));
        requiredList.add(Document.fromString("longitude"));

        // Create the root object
        Map<String, Document> rootMap = new HashMap<>();
        rootMap.put("type", Document.fromString("object"));
        rootMap.put("properties", Document.fromMap(propertiesMap));
        rootMap.put("required", Document.fromList(requiredList));

        // Now create the Document representing the JSON schema
        Document document = Document.fromMap(rootMap);

        ToolSpecification specification = ToolSpecification.builder()
            .name("Weather_Tool")
            .description("Get the current weather for a given location, based on its
WGS84 coordinates.")
            .inputSchema(ToolInputSchema.builder()
                .json(document)
                .build())
            .build();

        return specification;
    }

    /**
     * Fetches weather data for the given latitude and longitude.
     *
     * @param latitude  the latitude coordinate
     * @param longitude the longitude coordinate
     * @return a {@link CompletableFuture} containing the weather data as a JSON
string
     */
    public Document fetchWeatherData(String latitude, String longitude) {
        HttpClient httpClient = HttpClient.newHttpClient();
```

```
        // Ensure no extra double quotes
        latitude = latitude.replace("\"", "");
        longitude = longitude.replace("\"", "");

        String endpoint = "https://api.open-meteo.com/v1/forecast";
        String url = String.format("%s?latitude=%s&longitude=
%s&current_weather=True", endpoint, latitude, longitude);

        HttpRequest request = HttpRequest.newBuilder()
            .uri(URI.create(url))
            .build();

        try {
            HttpResponse<String> response = httpClient.send(request,
 HttpResponse.BodyHandlers.ofString());
            if (response.statusCode() == 200) {
                String weatherJson = response.body();
                System.out.println(weatherJson);
                ObjectMapper objectMapper = new ObjectMapper();
                Map<String, Object> rawMap = objectMapper.readValue(weatherJson, new
 TypeReference<Map<String, Object>>() {});
                Map<String, Document> documentMap = convertToDocumentMap(rawMap);


                Document weatherDocument = Document.fromMap(documentMap);
                System.out.println(weatherDocument);
                return weatherDocument;
            } else {
                throw new RuntimeException("Error fetching weather data: " +
 response.statusCode());
            }
        } catch (Exception e) {
            System.out.println("Error fetching weather data: " + e.getMessage());
            throw new RuntimeException("Error fetching weather data", e);
        }

    }

    private static Map<String, Document> convertToDocumentMap(Map<String, Object>
 inputMap) {
        Map<String, Document> result = new HashMap<>();
        for (Map.Entry<String, Object> entry : inputMap.entrySet()) {
            result.put(entry.getKey(), convertToDocument(entry.getValue()));
```

```
        }
        return result;
    }

    // Convert different types of Objects to Document
    private static Document convertToDocument(Object value) {
        if (value instanceof Map) {
            return Document.fromMap(convertToDocumentMap((Map<String, Object>)
 value));
        } else if (value instanceof Integer) {
            return Document.fromNumber(SdkNumber.fromInteger((Integer) value));
        } else if (value instanceof Double) {  //
            return Document.fromNumber(SdkNumber.fromDouble((Double) value));
        } else if (value instanceof Boolean) {
            return Document.fromBoolean((Boolean) value);
        } else if (value instanceof String) {
            return Document.fromString((String) value);
        }
        return Document.fromNull(); // Handle null values safely
    }
}
```

도구 구성이 포함된 Converse API 작업입니다.

```
    /**
     * Sends an asynchronous converse request to the AI model.
     *
     * @param modelId      the unique identifier of the AI model to be used for the
 converse request
     * @param systemPrompt the system prompt to be included in the converse request
     * @param conversation a list of messages representing the conversation history
     * @param toolSpec     the specification of the tool to be used in the converse
 request
     * @return the converse response received from the AI model
     */
    public ConverseResponse sendConverseRequestAsync(String modelId, String
 systemPrompt, List<Message> conversation, ToolSpecification toolSpec) {
        List<Tool> toolList = new ArrayList<>();
        Tool tool = Tool.builder()
            .toolSpec(toolSpec)
            .build();
```

```
        toolList.add(tool);

        ToolConfiguration configuration = ToolConfiguration.builder()
            .tools(toolList)
            .build();

        SystemContentBlock block = SystemContentBlock.builder()
            .text(systemPrompt)
            .build();

        ConverseRequest request = ConverseRequest.builder()
            .modelId(modelId)
            .system(block)
            .messages(conversation)
            .toolConfig(configuration)
            .build();

        try {
            ConverseResponse response = getClient().converse(request).join();
            return response;

        } catch (ModelNotReadyException ex) {
            throw new RuntimeException("Model is not ready: " + ex.getMessage(),
ex);
        } catch (BedrockRuntimeException ex) {
            throw new RuntimeException("Failed to converse with Bedrock model: " +
ex.getMessage(), ex);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Converse를 참조하세요.

Amazon Nova Canvas

InvokeModel

다음 코드 예제에서는 이미지 생성을 위해 Amazon Bedrock에서 Amazon Nova Canvas를 간접적으로 호출하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon Nova Canvas로 이미지를 생성합니다.

```
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 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelResponse;

import java.security.SecureRandom;
import java.util.Base64;

import static com.example.bedrockruntime.libs.ImageTools.displayImage;

/**
 * This example demonstrates how to use Amazon Nova Canvas to generate images.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Configure the image generation parameters
 * - Send a request to generate an image
 * - Process the response and handle the generated image
 */
public class InvokeModel {

    public static byte[] invokeModel() {

        // Step 1: Create the Amazon Bedrock runtime client
        // The runtime client handles the communication with AI models on Amazon
 Bedrock
        BedrockRuntimeClient client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
```

```
                .build();

        // Step 2: Specify which model to use
        // For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        String modelId = "amazon.nova-canvas-v1:0";

        // Step 3: Configure the generation parameters and create the request
        // First, set the main parameters:
        // - prompt: Text description of the image to generate
        // - seed: Random number for reproducible generation (0 to 858,993,459)
        String prompt = "A stylized picture of a cute old steampunk robot";
        int seed = new SecureRandom().nextInt(858_993_460);

        // Then, create the request using a template with the following structure:
        // - taskType: TEXT_IMAGE (specifies text-to-image generation)
        // - textToImageParams: Contains the text prompt
        // - imageGenerationConfig: Contains optional generation settings (seed,
 quality, etc.)
        // For a list of available request parameters, see:
        // https://docs.aws.amazon.com/nova/latest/userguide/image-gen-req-resp-
structure.html
        String request = """
                {
                    "taskType": "TEXT_IMAGE",
                    "textToImageParams": {
                        "text": "{{prompt}}"
                    },
                    "imageGenerationConfig": {
                        "seed": {{seed}},
                        "quality": "standard"
                    }
                }"""
                .replace("{{prompt}}", prompt)
                .replace("{{seed}}", String.valueOf(seed));

        // Step 4: Send and process the request
        // - Send the request to the model using InvokeModelResponse
        // - Extract the Base64-encoded image from the JSON response
        // - Convert the encoded image to a byte array and return it
        try {
            InvokeModelResponse response = client.invokeModel(builder -> builder
                    .modelId(modelId)
```

```
                    .body(SdkBytes.fromUtf8String(request))
            );

            JSONObject responseBody = new
 JSONObject(response.body().asUtf8String());
            // Convert the Base64 string to byte array for better handling
            return Base64.getDecoder().decode(
                    new JSONPointer("/images/0").queryFrom(responseBody).toString()
            );

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s%n", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        System.out.println("Generating image. This may take a few seconds...");
        byte[] imageData = invokeModel();
        displayImage(imageData);
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 InvokeModel을 참조하세요.

Amazon Titan Image Generator

InvokeModel

다음 코드 예제에서는 이미지 생성을 위해 Amazon Bedrock에서 Amazon Titan Image를 간접 호출하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon Titan Image Generator를 사용하여 이미지를 생성합니다.

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

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 InvokeModel을 참조하세요.

Amazon Titan Text Embeddings

InvokeModel

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 첫 번째 임베딩 생성을 시작합니다.
- 차원 수 및 정규화를 구성하는 임베딩을 생성합니다(V2만 해당).

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Titan Text Embeddings V2를 사용하여 첫 번째 임베딩을 생성합니다.

```
// 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();
    }
}
```

Titan Text Embeddings V2를 간접 호출하여 차원 수와 정규화를 구성합니다.

```
   /**
    * 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;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 InvokeModel을 참조하세요.

Anthropic Claude

Converse

다음 코드 예제에서는 Bedrock의 Converse API를 사용하여 Anthropic Claude에 텍스트 메시지를 보내는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Bedrock의 Converse API를 사용하여 Anthropic Claude에 텍스트 메시지를 보냅니다.

```
// 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().getFirst().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();
    }
}
```

Bedrock의 Converse API를 비동기 Java 클라이언트와 함께 사용하여 Anthropic Claude에 텍스트 메시지를 보냅니다.

```
// 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().getFirst().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();
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Converse를 참조하세요.

ConverseStream

다음 코드 예제에서는 Bedrock의 Converse API를 사용하여 Anthropic Claude에 텍스트 메시지를 보내고 응답 스트림을 실시간으로 처리하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Bedrock의 Converse API를 사용하여 Anthropic Claude에 텍스트 메시지를 보내고 응답 스트림을 실시간으로 처리합니다.

```
// 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());
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ConverseStream을 참조하세요.

InvokeModel

다음 코드 예제에서는 Invoke Model API를 사용하여 Anthropic Claude에 텍스트 메시지를 보내는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Invoke Model API를 사용하여 텍스트 메시지를 보냅니다.

```
// 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();
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 InvokeModel을 참조하세요.

InvokeModelWithResponseStream

다음 코드 예제에서는 Invoke Model API를 사용하여 Anthropic Claude 모델에 텍스트 메시지를 보내고 응답 스트림을 프린트하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Invoke Model API를 사용하여 텍스트 메시지를 보내고 응답 스트림을 실시간으로 처리합니다.

```
// 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() {

        // 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();
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 InvokeModelWithResponseStream을 참조하세요.

추론

다음 코드 예제에서는 Amazon Bedrock에서 Anthropic Claude 3.7 Sonnet의 추론 기능을 사용하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

비동기 Bedrock 런타임 클라이언트에서 Anthropic Claude 3.7 Sonnet의 추론 기능을 사용합니다.

```
import com.example.bedrockruntime.models.anthropicClaude.lib.ReasoningResponse;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
```

```
import software.amazon.awssdk.services.bedrockruntime.model.*;

import java.util.concurrent.CompletableFuture;

/**
 * This example demonstrates how to use Anthropic Claude 3.7 Sonnet's reasoning
 capability
 * with an asynchronous Amazon Bedrock runtime client.
 * It shows how to:
 * - Set up the Amazon Bedrock async runtime client
 * - Create a message
 * - Configure reasoning parameters
 * - Send an asynchronous request with reasoning enabled
 * - Process both the reasoning output and final response
 */
public class ReasoningAsync {

    public static ReasoningResponse reasoningAsync() {

        // Create the Amazon Bedrock runtime client
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        var modelId = "us.anthropic.claude-3-7-sonnet-20250219-v1:0";

        // Create the message with the user's prompt
        var prompt = "Describe the purpose of a 'hello world' program in one line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(prompt))
                .role(ConversationRole.USER)
                .build();

        // Configure reasoning parameters with a 2000 token budget
        Document reasoningConfig = Document.mapBuilder()
                .putDocument("thinking", Document.mapBuilder()
                        .putString("type", "enabled")
                        .putNumber("budget_tokens", 2000)
                        .build())
                .build();
```

```
        try {
            // Send message and reasoning configuration to the model
            CompletableFuture<ConverseResponse> asyncResponse =
client.converse(request -> request
                    .additionalModelRequestFields(reasoningConfig)
                    .messages(message)
                    .modelId(modelId)
            );

            // Process the response asynchronously
            return asyncResponse.thenApply(response -> {

                        var content = response.output().message().content();
                        ReasoningContentBlock reasoning = null;
                        String text = null;

                        // Process each content block to find reasoning and response
text
                        for (ContentBlock block : content) {
                            if (block.reasoningContent() != null) {
                                reasoning = block.reasoningContent();
                            } else if (block.text() != null) {
                                text = block.text();
                            }
                        }

                        return new ReasoningResponse(reasoning, text);
                    }
            ).get();

        } catch (Exception e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        // Execute the example and display reasoning and final response
        ReasoningResponse response = reasoningAsync();
        System.out.println("\n<thinking>");
        System.out.println(response.reasoning().reasoningText());
        System.out.println("</thinking>\n");
        System.out.println(response.text());
```

```
    }
}
```

동기 Bedrock 런타임 클라이언트에서 Anthropic Claude 3.7 Sonnet의 추론 기능을 사용합니다.

```
import com.example.bedrockruntime.models.anthropicClaude.lib.ReasoningResponse;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
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.*;

/**
 * This example demonstrates how to use Anthropic Claude 3.7 Sonnet's reasoning
 capability
 * with the synchronous Amazon Bedrock runtime client.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure reasoning parameters
 * - Send a request with reasoning enabled
 * - Process both the reasoning output and final response
 */
public class Reasoning {

    public static ReasoningResponse reasoning() {

        // Create the Amazon Bedrock runtime client
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        var modelId = "us.anthropic.claude-3-7-sonnet-20250219-v1:0";

        // Create the message with the user's prompt
        var prompt = "Describe the purpose of a 'hello world' program in one line.";
```

```
        var message = Message.builder()
                .content(ContentBlock.fromText(prompt))
                .role(ConversationRole.USER)
                .build();

        // Configure reasoning parameters with a 2000 token budget
        Document reasoningConfig = Document.mapBuilder()
                .putDocument("thinking", Document.mapBuilder()
                        .putString("type", "enabled")
                        .putNumber("budget_tokens", 2000)
                        .build())
                .build();

        try {
            // Send message and reasoning configuration to the model
            ConverseResponse bedrockResponse = client.converse(request -> request
                    .additionalModelRequestFields(reasoningConfig)
                    .messages(message)
                    .modelId(modelId)
            );


            // Extract both reasoning and final response
            var content = bedrockResponse.output().message().content();
            ReasoningContentBlock reasoning = null;
            String text = null;

            // Process each content block to find reasoning and response text
            for (ContentBlock block : content) {
                if (block.reasoningContent() != null) {
                    reasoning = block.reasoningContent();
                } else if (block.text() != null) {
                    text = block.text();
                }
            }

            return new ReasoningResponse(reasoning, 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) {
        // Execute the example and display reasoning and final response
        ReasoningResponse response = reasoning();
        System.out.println("\n<thinking>");
        System.out.println(response.reasoning().reasoningText());
        System.out.println("</thinking>\n");
        System.out.println(response.text());
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Converse를 참조하세요.

스트리밍 응답으로 추론

다음 코드 예제에서는 Amazon Bedrock에서 Anthropic Claude 3.7 Sonnet의 추론 기능을 사용하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Anthropic Claude 3.7 Sonnet의 추론 기능을 사용하여 스트리밍 텍스트 응답을 생성합니다.

```
import com.example.bedrockruntime.models.anthropicClaude.lib.ReasoningResponse;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

import java.util.concurrent.ExecutionException;
import java.util.concurrent.atomic.AtomicReference;

/**
```

```
 * This example demonstrates how to use Anthropic Claude 3.7 Sonnet's reasoning
 * capability to generate streaming text responses.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure a streaming request
 * - Set up a stream handler to process the response chunks
 * - Process the streaming response
 */
public class ReasoningStream {

    public static ReasoningResponse reasoningStream() {

        // Create the Amazon Bedrock runtime client
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        var modelId = "us.anthropic.claude-3-7-sonnet-20250219-v1:0";

        // Create the message with the user's prompt
        var prompt = "Describe the purpose of a 'hello world' program in one line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(prompt))
                .role(ConversationRole.USER)
                .build();

        // Configure reasoning parameters with a 2000 token budget
        Document reasoningConfig = Document.mapBuilder()
                .putDocument("thinking", Document.mapBuilder()
                        .putString("type", "enabled")
                        .putNumber("budget_tokens", 2000)
                        .build())
                .build();

        // Configure the request with the message, model ID, and reasoning config
        ConverseStreamRequest request = ConverseStreamRequest.builder()
                .additionalModelRequestFields(reasoningConfig)
                .messages(message)
                .modelId(modelId)
```

```
                .build();

        StringBuilder reasoning = new StringBuilder();
        StringBuilder text = new StringBuilder();
        AtomicReference<ReasoningResponse> finalresponse = new AtomicReference<>();

        // Set up the stream handler to processes chunks of the response as they
arrive
        var streamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            ContentBlockDelta delta = chunk.delta();
                            if (delta.reasoningContent() != null) {
                                if (reasoning.isEmpty()) {
                                    System.out.println("\n<thinking>");
                                }
                                if (delta.reasoningContent().text() != null) {

System.out.print(delta.reasoningContent().text());

reasoning.append(delta.reasoningContent().text());
                                }
                            } else if (delta.text() != null) {
                                if (text.isEmpty()) {
                                    System.out.println("\n</thinking>\n");
                                }
                                System.out.print(delta.text());
                                text.append(delta.text());
                            }
                            System.out.flush();  // Ensure immediate output of each
chunk
                        }).build())
                .onComplete(() -> finalresponse.set(new ReasoningResponse(
                        ReasoningContentBlock.fromReasoningText(t ->
t.text(reasoning.toString())),
                        text.toString()
                )))
                .onError(err -> System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage()))
                .build();

        // Step 6: Send the streaming request and process the response
        // - Send the request to the model
        // - Attach the handler to process response chunks as they arrive
```

```
        // - Handle any errors during streaming
        try {
            client.converseStream(request, streamHandler).get();
            return finalresponse.get();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
            throw new RuntimeException(e);
        } catch (Exception e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        reasoningStream();
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Converse를 참조하세요.

Cohere Command

Converse

다음 코드 예제에서는 Bedrock의 Converse API를 사용하여 Cohere Command에 텍스트 메시지를 보내는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Bedrock의 Converse API를 사용하여 Cohere Command로 텍스트 메시지를 보냅니다.

```
// 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();
    }
}
```

Bedrock의 Converse API를 비동기 Java 클라이언트와 함께 사용하여 Cohere Command로 텍스트 메시지를 보냅니다.

```
// 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();
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Converse를 참조하세요.

ConverseStream

다음 코드 예제에서는 Bedrock의 Converse API를 사용하여 Cohere Command에 텍스트 메시지를 보내고 응답 스트림을 실시간으로 처리하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Bedrock의 Converse API를 사용하여 Cohere Command에 텍스트 메시지를 보내고 응답 스트림을 실시간으로 처리합니다.

```
// 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());
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ConverseStream을 참조하세요.

InvokeModel: Command R 및 R+

다음 코드 예제에서는 Invoke Model API를 사용하여 Cohere Command R 및 R+에 텍스트 메시지를 보내는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Invoke Model API를 사용하여 텍스트 메시지를 보냅니다.

```
// 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();
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 InvokeModel을 참조하세요.

InvokeModelWithResponseStream: Command R 및 R+

다음 코드 예제에서는 Invoke Model API를 응답 스트림과 함께 사용하여 Cohere Command에 텍스트 메시지를 전송하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Invoke Model API를 사용하여 텍스트 메시지를 보내고 응답 스트림을 실시간으로 처리합니다.

```
// 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() {

        // 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();
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 InvokeModel을 참조하세요.

Meta Llama

Converse

다음 코드 예제에서는 Bedrock의 Converse API를 사용하여 Meta Llama에 텍스트 메시지를 보내는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Bedrock의 Converse API를 사용하여 Meta Llama에 텍스트 메시지를 보냅니다.

```
// 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();
    }
}
```

Bedrock의 Converse API를 비동기 Java 클라이언트와 함께 사용하여 Meta Llama에 텍스트 메시지를 보냅니다.

```
// 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();
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Converse를 참조하세요.

ConverseStream

다음 코드 예제에서는 Bedrock의 Converse API를 사용하여 Meta Llama에 텍스트 메시지를 보내고 응답 스트림을 실시간으로 처리하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Bedrock의 Converse API를 사용하여 Meta Llama에 텍스트 메시지를 보내고 응답 스트림을 실시간으로 처리합니다.

```
// 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());
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ConverseStream을 참조하세요.

InvokeModel

다음 코드 예제에서는 Invoke Model API를 사용하여 Meta Llama에 텍스트 메시지를 보내는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Invoke Model API를 사용하여 텍스트 메시지를 보냅니다.

```
// 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_WEST_2)
                .build();

        // Set the model ID, e.g., Llama 3 70b Instruct.
        var modelId = "meta.llama3-70b-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|><|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();
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 InvokeModel을 참조하세요.

InvokeModelWithResponseStream

다음 코드 예제에서는 Invoke Model API를 사용하여 Meta Llama에 텍스트 메시지를 보내고 응답 스트림을 프린트하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Invoke Model API를 사용하여 텍스트 메시지를 보내고 응답 스트림을 실시간으로 처리합니다.

```
// 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() {

        // 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_WEST_2)
                .build();

        // Set the model ID, e.g., Llama 3 70b Instruct.
        var modelId = "meta.llama3-70b-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|><|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();
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 InvokeModelWithResponseStream을 참조하세요.

Mistral AI

Converse

다음 코드 예제에서는 Bedrock의 Converse API를 사용하여 Mistral에 텍스트 메시지를 보내는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Bedrock의 Converse API를 사용하여 Mistral에 텍스트 메시지를 보냅니다.

```
// 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();
    }
}
```

Bedrock의 Converse API를 비동기 Java 클라이언트와 함께 사용하여 Mistral에 텍스트 메시지를 보냅니다.

```
// 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();
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Converse를 참조하세요.

ConverseStream

다음 코드 예제에서는 Bedrock의 Converse API를 사용하여 Mistral에 텍스트 메시지를 보내고 응답 스트림을 실시간으로 처리하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Bedrock의 Converse API를 사용하여 Mistral에 텍스트 메시지를 보내고 응답 스트림을 실시간으로 처리합니다.

```
// 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());
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ConverseStream을 참조하세요.

InvokeModel

다음 코드 예제에서는 Invoke Model API를 사용하여 Mistral 모델에 텍스트 메시지를 보내는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Invoke Model API를 사용하여 텍스트 메시지를 보냅니다.

```
// 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();
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 InvokeModel을 참조하세요.

InvokeModelWithResponseStream

다음 코드 예제에서는 Invoke Model API를 사용하여 Mistral AI 모델에 텍스트 메시지를 보내고 응답 스트림을 프린트하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Invoke Model API를 사용하여 텍스트 메시지를 보내고 응답 스트림을 실시간으로 처리합니다.

```
// 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() {
```

```
        // 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();
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 InvokeModelWithResponseStream을 참조하세요.

Stable Diffusion

InvokeModel

다음 코드 예제에서는 이미지 생성을 위해 Amazon Bedrock에서 Stability.ai Stable Diffusion XL 모델을 간접적으로 호출하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Stable Diffusion을 사용하여 이미지를 생성합니다.

```
// 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);
    }


}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 InvokeModel을 참조하세요.

Java 2.x용 SDK를 사용하는 CloudFront 예제

다음 코드 예제에서는 AWS SDK for Java 2.x CloudFront에서를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업
- 시나리오

작업

`CreateDistribution`

다음 코드 예시는 `CreateDistribution`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

다음 예제에서는 Amazon Simple Storage Service(Amazon S3) 버킷을 콘텐츠 오리진으로 사용합니다.

배포를 생성한 후 코드는 CloudFrontWaiter를 생성하여 배포가 완료될 때까지 기다린 후 배포를 반환합니다.

```
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;
        }
}
```

- API에 대한 세부 정보는AWS SDK for Java 2.x API 참조의 CreateDistribution을 참조하세요.

`CreateFunction`

다음 코드 예시는 `CreateFunction`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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 "";
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateFunction을 참조하세요.

`CreateKeyGroup`

다음 코드 예시는 `CreateKeyGroup`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

키 그룹에는 서명된 URL 또는 쿠키를 확인하는 데 사용되는 공개 키가 최소 하나 이상 필요합니다.

```
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;
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateKeyGroup을 참조하세요.

`CreatePublicKey`

다음 코드 예시는 `CreatePublicKey`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

다음 코드 예제는 퍼블릭 키를 읽고 Amazon CloudFront에 업로드합니다.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreatePublicKey을 참조하세요.

`DeleteDistribution`

다음 코드 예시는 `DeleteDistribution`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

다음 코드 예제는 배포를 비활성화됨으로 업데이트하고, 변경사항이 배포되기를 기다리는 웨이터를 사용한 다음 배포를 삭제합니다.

```
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);
                }
        }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteDistribution를 참조합니다.

`UpdateDistribution`

다음 코드 예시는 `UpdateDistribution`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateDistribution을 참조하세요.

시나리오

다중 테넌트 배포 및 배포 테넌트 생성

다음 코드 예제에서는 다양한 구성으로 다중 테넌트 배포 및 배포 테넌트를 생성하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

다음 예제에서는 파라미터와 와일드카드 인증서를 사용하여 다중 테넌트 배포를 생성하는 방법을 보여줍니다.

```
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.ConnectionMode;
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.HttpVersion;
import software.amazon.awssdk.services.cloudfront.model.Method;
import software.amazon.awssdk.services.cloudfront.model.SSLSupportMethod;
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 CreateMultiTenantDistribution {
    public static Distribution
 CreateMultiTenantDistributionWithCert(CloudFrontClient cloudFrontClient,
                                                                    S3Client
 s3Client,
                                                                    final String
 bucketName,
                                                                    final String
 certificateArn) {
        // fetch the origin info if necessary
        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.

        CreateDistributionResponse createDistResponse =
 cloudFrontClient.createDistribution(builder -> builder
                .distributionConfig(b1 -> b1
                        .httpVersion(HttpVersion.HTTP2)
                        .enabled(true)
                        .comment("Template Distribution with cert built with java")
                        .connectionMode(ConnectionMode.TENANT_ONLY)
```

```
                        .callerReference(Instant.now().toString())
                        .viewerCertificate(certBuilder -> certBuilder
                                .acmCertificateArn(certificateArn)
                                .sslSupportMethod(SSLSupportMethod.SNI_ONLY))
                        .origins(b2 -> b2
                                .quantity(1)
                                .items(b3 -> b3
                                        .domainName(originDomain)
                                        .id(originId)
                                        .originPath("/{{tenantName}}")
                                        .s3OriginConfig(builder4 -> builder4
                                                .originAccessIdentity(
                                                        ""))))
                        .tenantConfig(b5 -> b5
                                .parameterDefinitions(b6 -> b6
                                        .name("tenantName")
                                        .definition(b7 -> b7
                                                .stringSchema(b8 -> b8
                                                        .comment("tenantName value")
                                                        .defaultValue("root")
                                                        .required(false)))))
                        .defaultCacheBehavior(b2 -> b2

 .viewerProtocolPolicy(ViewerProtocolPolicy.ALLOW_ALL)
                                .targetOriginId(originId)
                                .cachePolicyId("658327ea-f89d-4fab-
a63d-7e88639e58f6") // CachingOptimized Policy
                                .allowedMethods(b4 -> b4
                                        .quantity(2)
                                        .items(Method.HEAD, Method.GET)))
                ));

        final Distribution distribution = createDistResponse.distribution();
        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"));
        }
```

```
        return distribution;
    }

    public static Distribution CreateMultiTenantDistributionNoCert(CloudFrontClient
cloudFrontClient,
                                                                   S3Client s3Client,
                                                                   final String
bucketName) {
        // fetch the origin info if necessary
        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.

        CreateDistributionResponse createDistResponse =
cloudFrontClient.createDistribution(builder -> builder
                .distributionConfig(b1 -> b1
                        .httpVersion(HttpVersion.HTTP2)
                        .enabled(true)
                        .comment("Template Distribution with cert built with java")
                        .connectionMode(ConnectionMode.TENANT_ONLY)
                        .callerReference(Instant.now().toString())
                        .origins(b2 -> b2
                                .quantity(1)
                                .items(b3 -> b3
                                        .domainName(originDomain)
                                        .id(originId)
                                        .originPath("/{{tenantName}}")
                                        .s3OriginConfig(builder4 -> builder4
                                                .originAccessIdentity(
                                                        ""))))
                        .tenantConfig(b5 -> b5
                                .parameterDefinitions(b6 -> b6
                                        .name("tenantName")
                                        .definition(b7 -> b7
                                                .stringSchema(b8 -> b8
                                                        .comment("tenantName value")
                                                        .defaultValue("root")
                                                        .required(false)))))
                        .defaultCacheBehavior(b2 -> b2

.viewerProtocolPolicy(ViewerProtocolPolicy.ALLOW_ALL)
```

```
                                .targetOriginId(originId)
                                .cachePolicyId("658327ea-f89d-4fab-
a63d-7e88639e58f6") // CachingOptimized Policy
                                .allowedMethods(b4 -> b4
                                        .quantity(2)
                                        .items(Method.HEAD, Method.GET)))
                ));

        final Distribution distribution = createDistResponse.distribution();
        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"));
        }
        return distribution;
    }
}
```

다음 예제에서는 위에서 선언한 파라미터를 사용하는 것을 포함하여 해당 템플릿과 연결된 배포 테넌트를 생성하는 방법을 보여줍니다. 도메인이 이미 상위 템플릿에 포함되어 있기 때문에 여기에 인증서 정보를 추가할 필요가 없습니다.

```
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.CreateConnectionGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionTenantResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionTenant;
import software.amazon.awssdk.services.cloudfront.model.GetConnectionGroupResponse;
import software.amazon.awssdk.services.cloudfront.model.ValidationTokenHost;
import software.amazon.awssdk.services.route53.Route53Client;
import software.amazon.awssdk.services.route53.model.RRType;

import java.time.Instant;
```

```
public class CreateDistributionTenant {

   public static DistributionTenant createDistributionTenantNoCert(CloudFrontClient
 cloudFrontClient,
                                                                    Route53Client
 route53Client,
                                                                    String
 distributionId,
                                                                    String domain,
                                                                    String
 hostedZoneId) {
       CreateDistributionTenantResponse createResponse =
 cloudFrontClient.createDistributionTenant(builder -> builder
               .distributionId(distributionId)
               .domains(b1 -> b1
                       .domain(domain))
               .parameters(b2 -> b2
                       .name("tenantName")
                       .value("myTenant"))
               .enabled(false)
               .name("no-cert-tenant")
       );

       final DistributionTenant distributionTenant =
 createResponse.distributionTenant();

       // Then update the Route53 hosted zone to point your domain at the
 distribution tenant
       // We fetch the RoutingEndpoint to point to via the default connection group
 that was created for your tenant
       final GetConnectionGroupResponse fetchedConnectionGroup =
 cloudFrontClient.getConnectionGroup(builder -> builder
               .identifier(distributionTenant.connectionGroupId()));

       route53Client.changeResourceRecordSets(builder -> builder
               .hostedZoneId(hostedZoneId)
               .changeBatch(b1 -> b1
                       .comment("ChangeBatch comment")
                       .changes(b2 -> b2
                               .resourceRecordSet(b3 -> b3
                                       .name(domain)
                                       .type("CNAME")
                                       .ttl(300L)
                                       .resourceRecords(b4 -> b4
```

```
 .value(fetchedConnectionGroup.connectionGroup().routingEndpoint())))
                                .action("CREATE"))
                ));
        return distributionTenant;
    }

}
```

뷰어 인증서가 상위 템플릿에서 생략된 경우 대신 연결된 테넌트(들)에 인증서 정보를 추가해야 합니다. 다음 예제에서는 테넌트에 필요한 도메인을 포함하는 ACM 인증서 ARN을 통해 이 작업을 수행하는 방법을 보여줍니다.

```
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.CreateConnectionGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionTenantResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionTenant;
import software.amazon.awssdk.services.cloudfront.model.GetConnectionGroupResponse;
import software.amazon.awssdk.services.cloudfront.model.ValidationTokenHost;
import software.amazon.awssdk.services.route53.Route53Client;
import software.amazon.awssdk.services.route53.model.RRType;

import java.time.Instant;

public class CreateDistributionTenant {

    public static DistributionTenant
 createDistributionTenantWithCert(CloudFrontClient cloudFrontClient,
                                                                    Route53Client
 route53Client,
                                                                    String
 distributionId,
                                                                    String domain,
                                                                    String
 hostedZoneId,
                                                                    String
 certificateArn) {
        CreateDistributionTenantResponse createResponse =
 cloudFrontClient.createDistributionTenant(builder -> builder
```

```
                .distributionId(distributionId)
                .domains(b1 -> b1
                        .domain(domain))
                .enabled(false)
                .name("tenant-with-cert")
                .parameters(b2 -> b2
                        .name("tenantName")
                        .value("myTenant"))
                .customizations(b3 -> b3
                        .certificate(b4 -> b4
                                .arn(certificateArn))) // NOTE: Cert must be in Us-
East-1 and cover the domain provided in this request

        );

        final DistributionTenant distributionTenant =
 createResponse.distributionTenant();

        // Then update the Route53 hosted zone to point your domain at the
 distribution tenant
        // We fetch the RoutingEndpoint to point to via the default connection group
 that was created for your tenant
        final GetConnectionGroupResponse fetchedConnectionGroup =
 cloudFrontClient.getConnectionGroup(builder -> builder
                .identifier(distributionTenant.connectionGroupId()));

        route53Client.changeResourceRecordSets(builder -> builder
                .hostedZoneId(hostedZoneId)
                .changeBatch(b1 -> b1
                        .comment("ChangeBatch comment")
                        .changes(b2 -> b2
                                .resourceRecordSet(b3 -> b3
                                        .name(domain)
                                        .type("CNAME")
                                        .ttl(300L)
                                        .resourceRecords(b4 -> b4

 .value(fetchedConnectionGroup.connectionGroup().routingEndpoint())))
                                .action("CREATE"))
                ));
        return distributionTenant;
    }

}
```

다음 예제에서는 CloudFront 호스팅 관리형 인증서 요청을 사용하여 이 작업을 수행하는 방법을 보여줍니다. 이는 도메인으로 향하는 트래픽이 아직 없는 경우에 적합합니다. 이 경우 ConnectionGroup을 만들어 RoutingEndpoint를 생성합니다. 그런 다음 해당 RoutingEndpoint를 사용하여 도메인 소유권을 확인하고 CloudFront를 가리키는 DNS 레코드를 생성합니다. 그러면 CloudFront가 토큰을 자동으로 제공하여 도메인 소유권을 검증하고 관리형 인증서를 생성합니다.

```
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.CreateConnectionGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionTenantResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionTenant;
import software.amazon.awssdk.services.cloudfront.model.GetConnectionGroupResponse;
import software.amazon.awssdk.services.cloudfront.model.ValidationTokenHost;
import software.amazon.awssdk.services.route53.Route53Client;
import software.amazon.awssdk.services.route53.model.RRType;

import java.time.Instant;

public class CreateDistributionTenant {

    public static DistributionTenant
 createDistributionTenantCfHosted(CloudFrontClient cloudFrontClient,
                                                                    Route53Client
 route53Client,
                                                                    String
 distributionId,
                                                                    String domain,
                                                                    String
 hostedZoneId) throws InterruptedException {
        CreateConnectionGroupResponse createConnectionGroupResponse =
 cloudFrontClient.createConnectionGroup(builder -> builder
                .ipv6Enabled(true)
                .name("cf-hosted-connection-group")
                .enabled(true));

        route53Client.changeResourceRecordSets(builder -> builder
                .hostedZoneId(hostedZoneId)
                .changeBatch(b1 -> b1
                        .comment("cf-hosted domain validation record")
```

```
                        .changes(b2 -> b2
                                .resourceRecordSet(b3 -> b3
                                        .name(domain)
                                        .type(RRType.CNAME)
                                        .ttl(300L)
                                        .resourceRecords(b4 -> b4

.value(createConnectionGroupResponse.connectionGroup().routingEndpoint())))
                                .action("CREATE"))
                ));

        // Give the R53 record time to propagate, if it isn't being returned by
servers yet, the following call will fail
        Thread.sleep(60000);

        CreateDistributionTenantResponse createResponse =
cloudFrontClient.createDistributionTenant(builder -> builder
                .distributionId(distributionId)
                .domains(b1 -> b1
                        .domain(domain))

.connectionGroupId(createConnectionGroupResponse.connectionGroup().id())
                .enabled(false)
                .name("cf-hosted-tenant")
                .parameters(b2 -> b2
                        .name("tenantName")
                        .value("myTenant"))
                .managedCertificateRequest(b3 -> b3
                        .validationTokenHost(ValidationTokenHost.CLOUDFRONT)
                )
        );

        return createResponse.distributionTenant();
    }

}
```

다음 예제에서는 자체 호스팅 관리형 인증서 요청을 사용하여 이 작업을 수행하는 방법을 보여 줍니다. 이는 도메인으로 향하는 트래픽이 있고 마이그레이션 중에 가동 중지 시간을 허용할 수 없는 경우에 적합합니다. 이 예제가 끝나면 도메인 검증 및 DNS 설정을 기다리는 상태로 테넌트가 생성됩니다. 트래픽을 마이그레이션할 준비가 되면 [여기](https://docs.aws.amazon.com/

AmazonCloudFront/latest/DeveloperGuide/managed-cloudfront-certificates.html#complete-domain-ownership)에 나온 단계에 따라 설정을 완료합니다.

```
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.CreateConnectionGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionTenantResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionTenant;
import software.amazon.awssdk.services.cloudfront.model.GetConnectionGroupResponse;
import software.amazon.awssdk.services.cloudfront.model.ValidationTokenHost;
import software.amazon.awssdk.services.route53.Route53Client;
import software.amazon.awssdk.services.route53.model.RRType;

import java.time.Instant;

public class CreateDistributionTenant {

    public static DistributionTenant
 createDistributionTenantSelfHosted(CloudFrontClient cloudFrontClient,
                                                                    String
 distributionId,
                                                                    String
 domain) {
        CreateDistributionTenantResponse createResponse =
 cloudFrontClient.createDistributionTenant(builder -> builder
                .distributionId(distributionId)
                .domains(b1 -> b1
                        .domain(domain))
                .parameters(b2 -> b2
                        .name("tenantName")
                        .value("myTenant"))
                .enabled(false)
                .name("self-hosted-tenant")
                .managedCertificateRequest(b3 -> b3
                        .validationTokenHost(ValidationTokenHost.SELF_HOSTED)
                        .primaryDomainName(domain)
                )
        );

        return createResponse.distributionTenant();
    }
```

```
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - CreateDistribution
  - CreateDistributionTenant

서명 리소스 삭제

다음 코드 예제는 Amazon Simple Storage Service(Amazon S3) 버킷의 제한된 콘텐츠에 액세스하는 데 사용되는 리소스를 삭제하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.

- DeleteKeyGroup
- DeleteOriginAccessControl
- DeletePublicKey

URL 및 쿠키에 서명

다음 코드 예제는 제한된 리소스에 액세스를 허용하는 서명된 URL 및 서명된 쿠키를 생성하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

CannedSignerRequest 클래스를 사용하면 미리 준비된 정책으로 URL 또는 쿠키에 서명할 수 있습니다.

```
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();
    }
}
```

CustomSignerRequest 클래스를 사용하면 사용자 지정 정책으로 URL 또는 쿠키에 서명할 수 있습니다. `activeDate`및 `ipRange`는 선택적 메서드입니다.

```
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)
                // .resourceUrlPattern("https://*.example.com/*")  // Optional.
                .privateKey(path)
```

```
                .keyPairId(publicKeyId)
                .expirationDate(expireDate)
                .activeDate(activeDate) // Optional.
                // .ipRange("192.168.0.1/24") // Optional.
                .build();
    }
}
```

다음 예제는 CloudFrontUtilities 클래스를 사용하여 서명된 쿠키와 URL을 생성하는 방법을 보여줍니다. GitHub에서 이 코드 예제를 확인하세요.

```
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;
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CloudFrontUtilities을 참조하세요.

Java 2.x용 SDK를 사용하는 CloudWatch 예제

다음 코드 예제에서는 CloudWatch와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시작하기

Hello CloudWatch

다음 코드 예제에서는 CloudWatch 사용을 시작하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListMetrics를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- CloudWatch 네임스페이스 및 지표를 나열합니다.
- 지표 및 예상 청구에 대한 통계를 가져옵니다.
- 대시보드를 생성하고 업데이트합니다.
- 데이터를 생성하여 지표에 추가합니다.
- 경보를 생성하고 트리거한 다음 경보 기록을 봅니다.
- 이상 탐지기를 추가합니다.
- 지표 이미지를 가져온 다음 리소스를 정리합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

CloudWatch 기능을 보여주는 대화형 시나리오를 실행합니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import
 software.amazon.awssdk.services.cloudwatch.model.DashboardInvalidInputErrorException;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsResponse;
import
 software.amazon.awssdk.services.cloudwatch.model.DeleteAnomalyDetectorResponse;
import software.amazon.awssdk.services.cloudwatch.model.DeleteDashboardsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricStatisticsResponse;
import software.amazon.awssdk.services.cloudwatch.model.LimitExceededException;
```

```
import software.amazon.awssdk.services.cloudwatch.model.PutDashboardResponse;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricDataResponse;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including 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", "-");
```

```
   static CloudWatchActions cwActions = new CloudWatchActions();

   private static final Logger logger =
LoggerFactory.getLogger(CloudWatchScenario.class);
   static Scanner scanner = new Scanner(System.in);
   public static void main(String[] args) throws Throwable {

       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/Billing 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 jsonWidgets.json in javav2/example_code/cloudwatch.)\s
             dashboardAdd - The location of a JSON file to use to update a
dashboard. (See CloudDashboard.json in javav2/example_code/cloudwatch.)\s
             settings - The location of a JSON file from which various values are
read. (See settings.json in javav2/example_code/cloudwatch.)\s
             metricImage - The location of a BMP file that is used to create a
graph.\s
           """;

       if (args.length != 7) {
           logger.info(usage);
           return;
       }
       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];

       logger.info(DASHES);
       logger.info("Welcome to the Amazon CloudWatch Basics scenario.");
       logger.info("""
```

```
            Amazon CloudWatch is a comprehensive monitoring and observability
service
            provided by Amazon Web Services (AWS). It is designed to help you
monitor your
            AWS resources, applications, and services, as well as on-premises
resources,
            in real-time.

            CloudWatch collects and tracks various types of data, including
metrics,
            logs, and events, from your AWS and on-premises resources. It allows you
to set
            alarms and automatically respond to changes in your environment,
            enabling you to quickly identify and address issues before they impact
your
            applications or services.

            With CloudWatch, you can gain visibility into your entire
infrastructure, from the cloud
            to the edge, and use this information to make informed decisions and
optimize your
            resource utilization.

            This scenario guides you through how to perform Amazon CloudWatch tasks
by using the
            AWS SDK for Java v2. Let's get started...
            """);
        waitForInputToContinue(scanner);

        try {
            runScenario(myDate, costDateWeek, dashboardName, dashboardJson,
dashboardAdd, settings, metricImage);
        } catch (RuntimeException e) {
            e.printStackTrace();
        }
        logger.info(DASHES);
    }

    private static void runScenario(String myDate, String costDateWeek, String
dashboardName, String dashboardJson, String dashboardAdd, String settings, String
metricImage ) throws Throwable {
        Double dataPoint = Double.parseDouble("10.0");
        logger.info(DASHES);
        logger.info("""
```

```
        1. List at least five available unique namespaces from Amazon CloudWatch.
        Select one from the list.
        """);
        String selectedNamespace;
        String selectedMetrics;
        int num;
        try {
            CompletableFuture<ArrayList<String>> future =
cwActions.listNameSpacesAsync();
            ArrayList<String> list = future.join();
            for (int z = 0; z < 5; z++) {
                int index = z + 1;
                logger.info("    " + index + ". {}", list.get(z));
            }

            num = Integer.parseInt(scanner.nextLine());
            if (1 <= num && num <= 5) {
                selectedNamespace = list.get(num - 1);
            } else {
                logger.info("You did not select a valid option.");
                return;
            }
            logger.info("You selected {}", selectedNamespace);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. List available metrics within the selected namespace.");
        logger.info("""
            A metric is a measure of the performance or health of your AWS
resources,
            applications, or custom resources. Metrics are the basic building blocks
of CloudWatch
```

```
            and provide data points that represent a specific aspect of your system
or application over time.

            Select a metric from the list.
            """);

        Dimension myDimension = null;
        try {
            CompletableFuture<ArrayList<String>> future =
cwActions.listMetsAsync(selectedNamespace);
            ArrayList<String> metList = future.join();
            logger.info("Metrics successfully retrieved. Total metrics: {}",
metList.size());
            for (int z = 0; z < 5; z++) {
                int index = z + 1;
                logger.info("    " + index + ". " + metList.get(z));
            }
            num = Integer.parseInt(scanner.nextLine());
            if (1 <= num && num <= 5) {
                selectedMetrics = metList.get(num - 1);
            } else {
                logger.info("You did not select a valid option.");
                return;
            }
            logger.info("You selected {}", selectedMetrics);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        try {
            myDimension = cwActions.getSpecificMetAsync(selectedNamespace).join();
            logger.info("Metric statistics successfully retrieved and displayed.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
```

```
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Get statistics for the selected metric over the last day.");
        logger.info("""
            Statistics refer to the various mathematical calculations that can be
performed on the
            collected metrics to derive meaningful insights. Statistics provide a
way to summarize and
            analyze the data collected for a specific metric over a specified time
period.
            """);
        waitForInputToContinue(scanner);
        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++) {
            logger.info("    " + (t + 1) + ". {}", statTypes.get(t));
        }
        logger.info("Select a metric statistic by entering a number from the
preceding list:");
        num = Integer.parseInt(scanner.nextLine());
        if (1 <= num && num <= 5) {
            metricOption = statTypes.get(num - 1);
        } else {
            logger.info("You did not select a valid option.");
            return;
        }
        logger.info("You selected " + metricOption);
        waitForInputToContinue(scanner);
```

```
        try {
            CompletableFuture<GetMetricStatisticsResponse> future =
cwActions.getAndDisplayMetricStatisticsAsync(selectedNamespace, selectedMetrics,
metricOption, myDate, myDimension);
            future.join();
            logger.info("Metric statistics retrieved successfully.");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Get CloudWatch estimated billing for the last week.");
        waitForInputToContinue(scanner);
         try {
            CompletableFuture<GetMetricStatisticsResponse> future =
cwActions.getMetricStatisticsAsync(costDateWeek);
            future.join();

            logger.info("Metric statistics successfully retrieved and displayed.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
             throw cause;
         }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Create a new CloudWatch dashboard with metrics.");
```

```
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutDashboardResponse> future =
cwActions.createDashboardWithMetricsAsync(dashboardName, dashboardJson);
            future.join();

        } catch (RuntimeException | IOException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof DashboardInvalidInputErrorException cwEx) {
                logger.info("Invalid CloudWatch data. Error message: {}, Error code
{}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. List dashboards using a paginator.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future = cwActions.listDashboardsAsync();
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Create a new custom metric by adding data to it.");
        logger.info("""
            The primary benefit of using a custom metric in Amazon CloudWatch is the
ability to
```

```
            monitor and collect data that is specific to your application or
infrastructure.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutMetricDataResponse> future =
cwActions.createNewCustomMetricAsync(dataPoint);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Add an additional metric to the dashboard.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutDashboardResponse> future =
cwActions.addMetricToDashboardAsync(dashboardAdd, dashboardName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof DashboardInvalidInputErrorException cwEx) {
                logger.info("Invalid CloudWatch data. Error message: {}, Error code
{}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("9. Create an alarm for the custom metric.");
```

```
        waitForInputToContinue(scanner);
        String alarmName = "" ;
        try {
            CompletableFuture<String> future = cwActions.createAlarmAsync(settings);
            alarmName = future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof LimitExceededException cwEx) {
                logger.info("The quota for alarms has been reached: Error message:
{}, Error code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("10. Describe ten current alarms.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future = cwActions.describeAlarmsAsync();
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("11. Get current data for new custom metric.");
        try {
            CompletableFuture<Void> future =
cwActions.getCustomMetricDataAsync(settings);
```

```
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("12. Push data into the custom metric to trigger the alarm.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutMetricDataResponse> future =
cwActions.addMetricDataForAlarmAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("13. Check the alarm state using the action
DescribeAlarmsForMetric.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
cwActions.checkForMetricAlarmAsync(settings);
            future.join();
```

```
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("14. Get alarm history for the new alarm.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
cwActions.getAlarmHistoryAsync(settings, myDate);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("15. Add an anomaly detector for the custom metric.");
        logger.info("""
            An anomaly detector is a feature that automatically detects unusual
patterns or deviations in your
            monitored metrics. It uses machine learning algorithms to analyze the
historical behavior
            of your metrics and establish a baseline.
```

```
            The anomaly detector then compares the current metric values against
this baseline and
            identifies any anomalies or outliers that may indicate potential issues
or unexpected changes
            in your system's performance or behavior.

            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
cwActions.addAnomalyDetectorAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("16. Describe current anomaly detectors.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
cwActions.describeAnomalyDetectorsAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
```

```
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("17. Get a metric image for the custom metric.");
        try {
            CompletableFuture<Void> future =
cwActions.downloadAndSaveMetricImageAsync(metricImage);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("18. Clean up the Amazon CloudWatch resources.");

        try {
            logger.info(". Delete the Dashboard.");
            waitForInputToContinue(scanner);
            CompletableFuture<DeleteDashboardsResponse> future =
cwActions.deleteDashboardAsync(dashboardName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        try {
```

```
            logger.info("Delete the alarm.");
            waitForInputToContinue(scanner);
            CompletableFuture<DeleteAlarmsResponse> future =
cwActions.deleteCWAlarmAsync(alarmName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        try {
            logger.info("Delete the anomaly detector.");
            waitForInputToContinue(scanner);
            CompletableFuture<DeleteAnomalyDetectorResponse> future =
cwActions.deleteAnomalyDetectorAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("The Amazon CloudWatch example scenario is complete.");
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
```

```
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();
            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

CloudWatch SDK 메서드의 래퍼 클래스입니다.

```
public class CloudWatchActions {

    private static CloudWatchAsyncClient cloudWatchAsyncClient;

    private static final Logger logger =
 LoggerFactory.getLogger(CloudWatchActions.class);

    /**
     * Retrieves an asynchronous CloudWatch client instance.
     *
     * <p>
     * This method ensures that the CloudWatch client is initialized with the
 following configurations:
     * <ul>
     *     <li>Maximum concurrency: 100</li>
     *     <li>Connection timeout: 60 seconds</li>
     *     <li>Read timeout: 60 seconds</li>
     *     <li>Write timeout: 60 seconds</li>
     *     <li>API call timeout: 2 minutes</li>
     *     <li>API call attempt timeout: 90 seconds</li>
     *     <li>Retry strategy: STANDARD</li>
     * </ul>
     * </p>
     *
     * @return the asynchronous CloudWatch client instance
```

```
     */
    private static CloudWatchAsyncClient getAsyncClient() {
        if (cloudWatchAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            cloudWatchAsyncClient = CloudWatchAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return cloudWatchAsyncClient;
    }

    /**
     * Deletes an Anomaly Detector.
     *
     * @param fileName the name of the file containing the Anomaly Detector
configuration
     * @return a CompletableFuture that represents the asynchronous deletion of the
Anomaly Detector
     */
    public CompletableFuture<DeleteAnomalyDetectorResponse>
deleteAnomalyDetectorAsync(String fileName) {
        CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                return new ObjectMapper().readTree(parser); // Return the root node
            } catch (IOException e) {
                throw new RuntimeException("Failed to read or parse the file", e);
            }
```

```
        });

        return readFileFuture.thenCompose(rootNode -> {
            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();

            return getAsyncClient().deleteAnomalyDetector(request);
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Failed to delete the Anomaly Detector",
exception);
            } else {
                logger.info("Successfully deleted the Anomaly Detector.");
            }
        });
    }

    /**
     * Deletes a CloudWatch alarm.
     *
     * @param alarmName the name of the alarm to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous operation
to delete the alarm
     * the {@link DeleteAlarmsResponse} is returned when the operation completes
successfully,
     * or a {@link RuntimeException} is thrown if the operation fails
     */
    public CompletableFuture<DeleteAlarmsResponse> deleteCWAlarmAsync(String
alarmName) {
        DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
```

```
            .alarmNames(alarmName)
            .build();

        return getAsyncClient().deleteAlarms(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to delete the alarm:{} " +
alarmName, exception);
                } else {
                    logger.info("Successfully deleted alarm {} ", alarmName);
                }
            });
    }

    /**
     * Deletes the specified dashboard.
     *
     * @param dashboardName the name of the dashboard to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous operation
of deleting the dashboard
     * @throws RuntimeException if the dashboard deletion fails
     */
    public CompletableFuture<DeleteDashboardsResponse> deleteDashboardAsync(String
dashboardName) {
        DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
            .dashboardNames(dashboardName)
            .build();

        return getAsyncClient().deleteDashboards(dashboardsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to delete the dashboard: " +
dashboardName, exception);
                } else {
                    logger.info("{} was successfully deleted.", dashboardName);
                }
            });
    }


    /**
     * Retrieves and saves a custom metric image to a file.
     *
```

```
    * @param fileName the name of the file to save the metric image to
    * @return a {@link CompletableFuture} that completes when the image has been
saved to the file
    */
   public CompletableFuture<Void> downloadAndSaveMetricImageAsync(String fileName)
{
       logger.info("Getting Image data for custom metric.");
       String myJSON = """
             {
                 "title": "Example Metric Graph",
                 "view": "timeSeries",
                 "stacked ": false,
                 "period": 10,
                 "width": 1400,
                 "height": 600,
                 "metrics": [
                     [
                     "AWS/Billing",
                     "EstimatedCharges",
                     "Currency",
                     "USD"
                    ]
                 ]
             }
            """;

       GetMetricWidgetImageRequest imageRequest =
GetMetricWidgetImageRequest.builder()
           .metricWidget(myJSON)
           .build();

       return getAsyncClient().getMetricWidgetImage(imageRequest)
           .thenCompose(response -> {
               SdkBytes sdkBytes = response.metricWidgetImage();
               byte[] bytes = sdkBytes.asByteArray();
               return CompletableFuture.runAsync(() -> {
                   try {
                       File outputFile = new File(fileName);
                       try (FileOutputStream outputStream = new
FileOutputStream(outputFile)) {
                           outputStream.write(bytes);
                       }
                   } catch (IOException e) {
```

```
                        throw new RuntimeException("Failed to write image to file",
e);
                    }
                });
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Error getting and saving metric
image", exception);
                } else {
                    logger.info("Image data saved successfully to {}", fileName);
                }
            });
    }


    /**
     * Describes the anomaly detectors based on the specified JSON file.
     *
     * @param fileName the name of the JSON file containing the custom metric
namespace and name
     * @return a {@link CompletableFuture} that completes when the anomaly detectors
have been described
     * @throws RuntimeException if there is a failure during the operation, such as
when reading or parsing the JSON file,
     *                          or when describing the anomaly detectors
     */
    public CompletableFuture<Void> describeAnomalyDetectorsAsync(String fileName) {
        CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                return new ObjectMapper().readTree(parser);
            } catch (IOException e) {
                throw new RuntimeException("Failed to read or parse the file", e);
            }
        });

        return readFileFuture.thenCompose(rootNode -> {
            try {
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
```

```
                String customMetricName =
rootNode.findValue("customMetricName").asText();

                DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
                    .maxResults(10)
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

                return
getAsyncClient().describeAnomalyDetectors(detectorsRequest).thenAccept(response ->
{
                    List<AnomalyDetector> anomalyDetectorList =
response.anomalyDetectors();
                    for (AnomalyDetector detector : anomalyDetectorList) {
                        logger.info("Metric name: {} ",
detector.singleMetricAnomalyDetector().metricName());
                        logger.info("State: {} ", detector.stateValue());
                    }
                });
            } catch (RuntimeException e) {
                throw new RuntimeException("Failed to describe anomaly detectors",
e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error describing anomaly detectors",
exception);
            }
        });
    }


    /**
     * Adds an anomaly detector for the given file.
     *
     * @param fileName the name of the file containing the anomaly detector
configuration
     * @return a {@link CompletableFuture} that completes when the anomaly detector
has been added
     */
    public CompletableFuture<Void> addAnomalyDetectorAsync(String fileName) {
```

```
        CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                return new ObjectMapper().readTree(parser); // Return the root node
            } catch (IOException e) {
                throw new RuntimeException("Failed to read or parse the file", e);
            }
        });

        return readFileFuture.thenCompose(rootNode -> {
            try {
                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();

                return
getAsyncClient().putAnomalyDetector(anomalyDetectorRequest).thenAccept(response ->
{
                    logger.info("Added anomaly detector for metric {}",
customMetricName);
                });
            } catch (Exception e) {
                throw new RuntimeException("Failed to create anomaly detector", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error adding anomaly detector",
exception);
            }
```

```
        });
    }


    /**
     * Retrieves the alarm history for a given alarm name and date range.
     *
     * @param fileName the path to the JSON file containing the alarm name
     * @param date     the date to start the alarm history search (in the format
"yyyy-MM-dd'T'HH:mm:ss'Z'")
     * @return a {@code CompletableFuture<Void>} that completes when the alarm
history has been retrieved and processed
     */
    public CompletableFuture<Void> getAlarmHistoryAsync(String fileName, String
date) {
        CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
                return rootNode.findValue("exampleAlarmName").asText(); // Return
alarmName from the JSON file
            } catch (IOException e) {
                throw new RuntimeException("Failed to read or parse the file", e);
            }
        });

        // Use the alarm name to describe alarm history with a paginator.
        return readFileFuture.thenCompose(alarmName -> {
            try {
                Instant start = Instant.parse(date);
                Instant endDate = Instant.now();
                DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
                    .startDate(start)
                    .endDate(endDate)
                    .alarmName(alarmName)
                    .historyItemType(HistoryItemType.ACTION)
                    .build();

                // Use the paginator to paginate through alarm history pages.
```

```
                DescribeAlarmHistoryPublisher historyPublisher =
getAsyncClient().describeAlarmHistoryPaginator(historyRequest);
                CompletableFuture<Void> future = historyPublisher
                    .subscribe(response -> response.alarmHistoryItems().forEach(item
-> {
                        logger.info("History summary: {}", item.historySummary());
                        logger.info("Timestamp: {}", item.timestamp());
                    }))
                    .whenComplete((result, exception) -> {
                        if (exception != null) {
                            logger.error("Error occurred while getting alarm
history: " + exception.getMessage(), exception);
                        } else {
                            logger.info("Successfully retrieved all alarm
history.");
                        }
                    });

                // Return the future to the calling code for further handling
                return future;
            } catch (Exception e) {
                throw new RuntimeException("Failed to process alarm history", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error completing alarm history
processing", exception);
            }
        });
    }



    /**
     * Checks for a metric alarm in AWS CloudWatch.
     *
     * @param fileName the name of the file containing the JSON configuration for
the custom metric
     * @return a {@link CompletableFuture} that completes when the check for the
metric alarm is complete
     */
    public CompletableFuture<Void> checkForMetricAlarmAsync(String fileName) {
        CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
```

```
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
                return rootNode.toString(); // Return JSON as a string for further
processing
            } catch (IOException e) {
                throw new RuntimeException("Failed to read file", e);
            }
        });

        return readFileFuture.thenCompose(jsonContent -> {
            try {
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();

                DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

                return checkForAlarmAsync(metricRequest, customMetricName, 10);

            } catch (IOException e) {
                throw new RuntimeException("Failed to parse JSON content", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error checking metric alarm",
exception);
            }
        });
    }

    // Recursive method to check for the alarm.

    /**
     * Checks for the existence of an alarm asynchronously for the specified metric.
```

```
    *
    * @param metricRequest    the request to describe the alarms for the specified
metric
    * @param customMetricName the name of the custom metric to check for an alarm
    * @param retries          the number of retries to perform if no alarm is found
    * @return a {@link CompletableFuture} that completes when an alarm is found or
the maximum number of retries has been reached
    */
   private static CompletableFuture<Void>
checkForAlarmAsync(DescribeAlarmsForMetricRequest metricRequest, String
customMetricName, int retries) {
       if (retries == 0) {
           return CompletableFuture.completedFuture(null).thenRun(() ->
               logger.info("No Alarm state found for {} after 10 retries.",
customMetricName)
           );
       }

       return
(getAsyncClient().describeAlarmsForMetric(metricRequest).thenCompose(response -> {
           if (response.hasMetricAlarms()) {
               logger.info("Alarm state found for {}", customMetricName);
               return CompletableFuture.completedFuture(null); // Alarm found,
complete the future
           } else {
               return CompletableFuture.runAsync(() -> {
                   try {
                       Thread.sleep(20000);
                       logger.info(".");
                   } catch (InterruptedException e) {
                       throw new RuntimeException("Interrupted while waiting to
retry", e);
                   }
               }).thenCompose(v -> checkForAlarmAsync(metricRequest,
customMetricName, retries - 1)); // Recursive call
           }
       }));
   }


   /**
    * Adds metric data for an alarm asynchronously.
    *
    * @param fileName the name of the JSON file containing the metric data
```

```
    * @return a CompletableFuture that asynchronously returns the
PutMetricDataResponse
    */
   public CompletableFuture<PutMetricDataResponse>
addMetricDataForAlarmAsync(String fileName) {
       CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
               return rootNode.toString(); // Return JSON as a string for further
processing
           } catch (IOException e) {
               throw new RuntimeException("Failed to read file", e);
           }
       });

       return readFileFuture.thenCompose(jsonContent -> {
           try {
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
               String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
               String customMetricName =
rootNode.findValue("customMetricName").asText();
               Instant instant = Instant.now();

               // Create MetricDatum objects.
               MetricDatum datum1 = 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(datum1);
                metricDataList.add(datum2);

                // Build the PutMetricData request.
                PutMetricDataRequest request = PutMetricDataRequest.builder()
                    .namespace(customMetricNamespace)
                    .metricData(metricDataList)
                    .build();

                // Send the request asynchronously.
                return getAsyncClient().putMetricData(request);

            } catch (IOException e) {
                CompletableFuture<PutMetricDataResponse> failedFuture = new
CompletableFuture<>();
                failedFuture.completeExceptionally(new RuntimeException("Failed to
parse JSON content", e));
                return failedFuture;
            }
        }).whenComplete((response, exception) -> {
            if (exception != null) {
                logger.error("Failed to put metric data: " + exception.getMessage(),
exception);
            } else {
                logger.info("Added metric values for metric.");
            }
        });
    }


    /**
     * Retrieves custom metric data from the AWS CloudWatch service.
     *
     * @param fileName the name of the file containing the custom metric information
     * @return a {@link CompletableFuture} that completes when the metric data has
been retrieved
     */
    public CompletableFuture<Void> getCustomMetricDataAsync(String fileName) {
        CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
            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);
                return rootNode.toString(); // Return JSON as a string for further
processing
            } catch (IOException e) {
                throw new RuntimeException("Failed to read file", e);
            }
        });

        return readFileFuture.thenCompose(jsonContent -> {
            try {
                // Parse the JSON string to extract relevant values.
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();

                // Set the current time and date range for metric query.
                Instant nowDate = Instant.now();
                long hours = 1;
                long minutes = 30;
                Instant endTime = 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(60)  // Assuming period in seconds
                    .metric(met)
                    .build();

                MetricDataQuery dataQuery = MetricDataQuery.builder()
                    .metricStat(metStat)
                    .id("foo2")
                    .returnData(true)
                    .build();
```

```
                List<MetricDataQuery> dq = new ArrayList<>();
                dq.add(dataQuery);

                GetMetricDataRequest getMetricDataRequest =
GetMetricDataRequest.builder()
                    .maxDatapoints(10)
                    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
                    .startTime(nowDate)
                    .endTime(endTime)
                    .metricDataQueries(dq)
                    .build();

                // Call the async method for CloudWatch data retrieval.
                return getAsyncClient().getMetricData(getMetricDataRequest);

            } catch (IOException e) {
                throw new RuntimeException("Failed to parse JSON content", e);
            }
        }).thenAccept(response -> {
            List<MetricDataResult> data = response.metricDataResults();
            for (MetricDataResult item : data) {
                logger.info("The label is: {}", item.label());
                logger.info("The status code is: {}", item.statusCode().toString());
            }
        }).exceptionally(exception -> {
            throw new RuntimeException("Failed to get metric data", exception);
        });
    }


    /**
     * Describes the CloudWatch alarms of the 'METRIC_ALARM' type.
     *
     * @return a {@link CompletableFuture} that represents the asynchronous
operation
     * of describing the CloudWatch alarms. The future completes when the
     * operation is finished, either successfully or with an error.
     */
    public CompletableFuture<Void> describeAlarmsAsync() {
        List<AlarmType> typeList = new ArrayList<>();
        typeList.add(AlarmType.METRIC_ALARM);
        DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
            .alarmTypes(typeList)
```

```
            .maxRecords(10)
            .build();

        return getAsyncClient().describeAlarms(alarmsRequest)
            .thenAccept(response -> {
                List<MetricAlarm> alarmList = response.metricAlarms();
                for (MetricAlarm alarm : alarmList) {
                    logger.info("Alarm name: {}", alarm.alarmName());
                    logger.info("Alarm description: {} ", alarm.alarmDescription());
                }
            })
            .whenComplete((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to describe alarms: {}", ex.getMessage());
                } else {
                    logger.info("Successfully described alarms.");
                }
            });
    }

    /**
     * Creates an alarm based on the configuration provided in a JSON file.
     *
     * @param fileName the name of the JSON file containing the alarm configuration
     * @return a CompletableFuture that represents the asynchronous operation of
creating the alarm
     * @throws RuntimeException if an exception occurs while reading the JSON file
or creating the alarm
     */
    public CompletableFuture<String> createAlarmAsync(String fileName) {
        com.fasterxml.jackson.databind.JsonNode rootNode;
        try {
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            rootNode = new ObjectMapper().readTree(parser);
        } catch (IOException e) {
            throw new RuntimeException("Failed to read the alarm configuration
file", e);
        }

        // Extract values from the JSON node.
        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();

        // Call the putMetricAlarm asynchronously and handle the result.
        return getAsyncClient().putMetricAlarm(alarmRequest)
            .handle((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to create alarm: {}", ex.getMessage());
                    throw new RuntimeException("Failed to create alarm", ex);
                } else {
                    logger.info("{} was successfully created!", alarmName);
                    return alarmName;
                }
            });
    }

    /**
     * Adds a metric to a dashboard asynchronously.
     *
     * @param fileName      the name of the file containing the dashboard content
     * @param dashboardName the name of the dashboard to be updated
```

```
    * @return a {@link CompletableFuture} representing the asynchronous operation,
which will complete with a
    * {@link PutDashboardResponse} when the dashboard is successfully updated
    */
   public CompletableFuture<PutDashboardResponse> addMetricToDashboardAsync(String
fileName, String dashboardName) {
       String dashboardBody;
       try {
           dashboardBody = readFileAsString(fileName);
       } catch (IOException e) {
           throw new RuntimeException("Failed to read the dashboard file", e);
       }

       PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
           .dashboardName(dashboardName)
           .dashboardBody(dashboardBody)
           .build();

       return getAsyncClient().putDashboard(dashboardRequest)
           .handle((response, ex) -> {
               if (ex != null) {
                   logger.info("Failed to update dashboard: {}", ex.getMessage());
                   throw new RuntimeException("Error updating dashboard", ex);
               } else {
                   logger.info("{} was successfully updated.", dashboardName);
                   return response;
               }
           });
   }

   /**
    * Creates a new custom metric.
    *
    * @param dataPoint the data point to be added to the custom metric
    * @return a {@link CompletableFuture} representing the asynchronous operation
of adding the custom metric
    */
   public CompletableFuture<PutMetricDataResponse>
createNewCustomMetricAsync(Double dataPoint) {
       Dimension dimension = Dimension.builder()
           .name("UNIQUE_PAGES")
           .value("URLS")
           .build();
```

```
        // Set an Instant object for the current time in UTC.
        String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
        Instant instant = Instant.parse(time);

        // Create the MetricDatum.
        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();

        return getAsyncClient().putMetricData(request)
            .whenComplete((response, ex) -> {
                if (ex != null) {
                    throw new RuntimeException("Error adding custom metric", ex);
                } else {
                    logger.info("Successfully added metric values for
PAGES_VISITED.");
                }
            });
    }

    /**
     * Lists the available dashboards.
     *
     * @return a {@link CompletableFuture} that completes when the operation is
finished.
     * The future will complete exceptionally if an error occurs while listing the
dashboards.
     */
    public CompletableFuture<Void> listDashboardsAsync() {
        ListDashboardsRequest listDashboardsRequest =
ListDashboardsRequest.builder().build();
        ListDashboardsPublisher paginator =
getAsyncClient().listDashboardsPaginator(listDashboardsRequest);
        return paginator.subscribe(response -> {
```

```
            response.dashboardEntries().forEach(entry -> {
                logger.info("Dashboard name is: {} ", entry.dashboardName());
                logger.info("Dashboard ARN is: {} ", entry.dashboardArn());
            });
        }).exceptionally(ex -> {
            logger.info("Failed to list dashboards: {} ", ex.getMessage());
            throw new RuntimeException("Error occurred while listing dashboards",
ex);
        });
    }


    /**
     * Creates a new dashboard with the specified name and metrics from the given
file.
     *
     * @param dashboardName the name of the dashboard to be created
     * @param fileName      the name of the file containing the dashboard body
     * @return a {@link CompletableFuture} representing the asynchronous operation
of creating the dashboard
     * @throws IOException if there is an error reading the dashboard body from the
file
     */
    public CompletableFuture<PutDashboardResponse>
createDashboardWithMetricsAsync(String dashboardName, String fileName) throws
IOException {
        String dashboardBody = readFileAsString(fileName);
        PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
            .dashboardName(dashboardName)
            .dashboardBody(dashboardBody)
            .build();

        return getAsyncClient().putDashboard(dashboardRequest)
            .handle((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to create dashboard: {}", ex.getMessage());
                    throw new RuntimeException("Dashboard creation failed", ex);
                } else {
                    // Handle the normal response case
                    logger.info("{} was successfully created.", dashboardName);
                    List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
                    if (messages.isEmpty()) {
                        logger.info("There are no messages in the new Dashboard.");
```

```
                    } else {
                        for (DashboardValidationMessage message : messages) {
                            logger.info("Message: {}", message.message());
                        }
                    }
                    return response; // Return the response for further use
                }
            });
    }


    /**
     * Retrieves the metric statistics for the "EstimatedCharges" metric in the
"AWS/Billing" namespace.
     *
     * @param costDateWeek the start date for the metric statistics, in the format
of an ISO-8601 date string (e.g., "2023-04-05")
     * @return a {@link CompletableFuture} that, when completed, contains the {@link
GetMetricStatisticsResponse} with the retrieved metric statistics
     * @throws RuntimeException if the metric statistics cannot be retrieved
successfully
     */
    public CompletableFuture<GetMetricStatisticsResponse>
getMetricStatisticsAsync(String costDateWeek) {
        Instant start = Instant.parse(costDateWeek);
        Instant endDate = Instant.now();

        // Define dimension
        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) // One day period
            .build();

        return getAsyncClient().getMetricStatistics(statisticsRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    List<Datapoint> data = response.datapoints();
                    if (!data.isEmpty()) {
                        for (Datapoint datapoint : data) {
                            logger.info("Timestamp: {} Maximum value: {})",
datapoint.timestamp(), datapoint.maximum());
                        }
                    } else {
                        logger.info("The returned data list is empty");
                    }
                } else {
                    throw new RuntimeException("Failed to get metric statistics: " +
exception.getMessage(), exception);
                }
            });
    }


    /**
     * Retrieves and displays metric statistics for the specified parameters.
     *
     * @param nameSpace    the namespace for the metric
     * @param metVal       the name of the metric
     * @param metricOption the statistic to retrieve for the metric (e.g.,
"Maximum", "Average")
     * @param date         the date for which to retrieve the metric statistics, in
the format "yyyy-MM-dd'T'HH:mm:ss'Z'"
     * @param myDimension  the dimension(s) to filter the metric statistics by
     * @return a {@link CompletableFuture} that completes when the metric statistics
have been retrieved and displayed
     */
    public CompletableFuture<GetMetricStatisticsResponse>
getAndDisplayMetricStatisticsAsync(String nameSpace, String metVal,

         String metricOption, String date, Dimension myDimension) {

        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();
```

```
        // Building the request for metric statistics.
        GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
            .endTime(endDate)
            .startTime(start)
            .dimensions(myDimension)
            .metricName(metVal)
            .namespace(nameSpace)
            .period(86400) // 1 day period
            .statistics(Statistic.fromValue(metricOption))
            .build();

        return getAsyncClient().getMetricStatistics(statisticsRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    List<Datapoint> data = response.datapoints();
                    if (!data.isEmpty()) {
                        for (Datapoint datapoint : data) {
                            logger.info("Timestamp: {} Maximum value: {}",
datapoint.timestamp(), datapoint.maximum());
                        }
                    } else {
                        logger.info("The returned data list is empty");
                    }
                } else {
                    logger.info("Failed to get metric statistics: {} ",
exception.getMessage());
                }
            })
            .exceptionally(exception -> {
                throw new RuntimeException("Error while getting metric statistics: "
+ exception.getMessage(), exception);
            });
    }


    /**
     * Retrieves a list of metric names for the specified namespace.
     *
     * @param namespace the namespace for which to retrieve the metric names
     * @return a {@link CompletableFuture} that, when completed, contains an {@link
ArrayList} of
     * the metric names in the specified namespace
     * @throws RuntimeException if an error occurs while listing the metrics
```

```
     */
    public CompletableFuture<ArrayList<String>> listMetsAsync(String namespace) {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        ListMetricsPublisher metricsPaginator =
getAsyncClient().listMetricsPaginator(request);
        Set<String> metSet = new HashSet<>();
        CompletableFuture<Void> future = metricsPaginator.subscribe(response -> {
            response.metrics().forEach(metric -> {
                String metricName = metric.metricName();
                metSet.add(metricName);
            });
        });

        return future
            .thenApply(ignored -> new ArrayList<>(metSet))
            .exceptionally(exception -> {
                throw new RuntimeException("Failed to list metrics: " +
exception.getMessage(), exception);
            });
    }

    /**
     * Lists the available namespaces for the current AWS account.
     *
     * @return a {@link CompletableFuture} that, when completed, contains an {@link
ArrayList} of the available namespace names.
     * @throws RuntimeException if an error occurs while listing the namespaces.
     */
    public CompletableFuture<ArrayList<String>> listNameSpacesAsync() {
        ArrayList<String> nameSpaceList = new ArrayList<>();
        ListMetricsRequest request = ListMetricsRequest.builder().build();

        ListMetricsPublisher metricsPaginator =
getAsyncClient().listMetricsPaginator(request);
        CompletableFuture<Void> future = metricsPaginator.subscribe(response -> {
            response.metrics().forEach(metric -> {
                String namespace = metric.namespace();
                if (!nameSpaceList.contains(namespace)) {
                    nameSpaceList.add(namespace);
                }
            });
```

```
        });

        return future
            .thenApply(ignored -> nameSpaceList)
            .exceptionally(exception -> {
                throw new RuntimeException("Failed to list namespaces: " +
exception.getMessage(), exception);
            });
    }
    /**
     * Retrieves the specific metric asynchronously.
     *
     * @param namespace the namespace of the metric to retrieve
     * @return a CompletableFuture that completes with the first dimension of the
first metric found in the specified namespace,
     * or throws a RuntimeException if an error occurs or no metrics or dimensions
are found
     */
    public CompletableFuture<Dimension> getSpecificMetAsync(String namespace) {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        return getAsyncClient().listMetrics(request).handle((response, exception) ->
{
            if (exception != null) {
                logger.info("Error occurred while listing metrics: {} ",
exception.getMessage());
                throw new RuntimeException("Failed to retrieve specific metric
dimension", exception);
            } else {
                List<Metric> myList = response.metrics();
                if (!myList.isEmpty()) {
                    Metric metric = myList.get(0);
                    if (!metric.dimensions().isEmpty()) {
                        return metric.dimensions().get(0); // Return the first
dimension
                    }
                }
                throw new RuntimeException("No metrics or dimensions found");
            }
        });
    }
```

```
    public static String readFileAsString(String file) throws IOException {
        return new String(Files.readAllBytes(Paths.get(file)));
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 항목을 참조하세요.
  - DeleteAlarms
  - DeleteAnomalyDetector
  - DeleteDashboards
  - DescribeAlarmHistory
  - DescribeAlarms
  - DescribeAlarmsForMetric
  - DescribeAnomalyDetectors
  - GetMetricData
  - GetMetricStatistics
  - GetMetricWidgetImage
  - ListMetrics
  - PutAnomalyDetector
  - PutDashboard
  - PutMetricAlarm
  - PutMetricData

작업

`DeleteAlarms`

다음 코드 예시는 `DeleteAlarms`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Deletes a CloudWatch alarm.
     *
     * @param alarmName the name of the alarm to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous operation
to delete the alarm
     * the {@link DeleteAlarmsResponse} is returned when the operation completes
successfully,
     * or a {@link RuntimeException} is thrown if the operation fails
     */
    public CompletableFuture<DeleteAlarmsResponse> deleteCWAlarmAsync(String
alarmName) {
        DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
            .alarmNames(alarmName)
            .build();

        return getAsyncClient().deleteAlarms(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to delete the alarm:{} " +
alarmName, exception);
                } else {
                    logger.info("Successfully deleted alarm {} ", alarmName);
                }
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteAlarms를 참조하세요.

`DeleteAnomalyDetector`

다음 코드 예시는 `DeleteAnomalyDetector`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Deletes an Anomaly Detector.
    *
    * @param fileName the name of the file containing the Anomaly Detector
configuration
    * @return a CompletableFuture that represents the asynchronous deletion of the
Anomaly Detector
    */
   public CompletableFuture<DeleteAnomalyDetectorResponse>
deleteAnomalyDetectorAsync(String fileName) {
       CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               return new ObjectMapper().readTree(parser); // Return the root node
           } catch (IOException e) {
               throw new RuntimeException("Failed to read or parse the file", e);
           }
       });

       return readFileFuture.thenCompose(rootNode -> {
           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();

           return getAsyncClient().deleteAnomalyDetector(request);
       }).whenComplete((result, exception) -> {
           if (exception != null) {
```

```
                throw new RuntimeException("Failed to delete the Anomaly Detector",
exception);
            } else {
                logger.info("Successfully deleted the Anomaly Detector.");
            }
        });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteAnomalyDetector를 참조하세요.

`DeleteDashboards`

다음 코드 예시는 `DeleteDashboards`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Deletes the specified dashboard.
     *
     * @param dashboardName the name of the dashboard to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous operation
of deleting the dashboard
     * @throws RuntimeException if the dashboard deletion fails
     */
    public CompletableFuture<DeleteDashboardsResponse> deleteDashboardAsync(String
dashboardName) {
        DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
            .dashboardNames(dashboardName)
            .build();

        return getAsyncClient().deleteDashboards(dashboardsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
```

```
                    throw new RuntimeException("Failed to delete the dashboard: " +
dashboardName, exception);
                } else {
                    logger.info("{} was successfully deleted.", dashboardName);
                }
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteDashboards를 참조하세요.

`DescribeAlarmHistory`

다음 코드 예시는 `DescribeAlarmHistory`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Retrieves the alarm history for a given alarm name and date range.
     *
     * @param fileName the path to the JSON file containing the alarm name
     * @param date     the date to start the alarm history search (in the format
"yyyy-MM-dd'T'HH:mm:ss'Z'")
     * @return a {@code CompletableFuture<Void>} that completes when the alarm
history has been retrieved and processed
     */
    public CompletableFuture<Void> getAlarmHistoryAsync(String fileName, String
date) {
        CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
```

```
                return rootNode.findValue("exampleAlarmName").asText(); // Return
alarmName from the JSON file
            } catch (IOException e) {
                throw new RuntimeException("Failed to read or parse the file", e);
            }
        });

        // Use the alarm name to describe alarm history with a paginator.
        return readFileFuture.thenCompose(alarmName -> {
            try {
                Instant start = Instant.parse(date);
                Instant endDate = Instant.now();
                DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
                    .startDate(start)
                    .endDate(endDate)
                    .alarmName(alarmName)
                    .historyItemType(HistoryItemType.ACTION)
                    .build();

                // Use the paginator to paginate through alarm history pages.
                DescribeAlarmHistoryPublisher historyPublisher =
getAsyncClient().describeAlarmHistoryPaginator(historyRequest);
                CompletableFuture<Void> future = historyPublisher
                    .subscribe(response -> response.alarmHistoryItems().forEach(item
-> {
                        logger.info("History summary: {}", item.historySummary());
                        logger.info("Timestamp: {}", item.timestamp());
                    }))
                    .whenComplete((result, exception) -> {
                        if (exception != null) {
                            logger.error("Error occurred while getting alarm
history: " + exception.getMessage(), exception);
                        } else {
                            logger.info("Successfully retrieved all alarm
history.");
                        }
                    });

                // Return the future to the calling code for further handling
                return future;
            } catch (Exception e) {
                throw new RuntimeException("Failed to process alarm history", e);
            }
```

```
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error completing alarm history
processing", exception);
            }
        });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeAlarmHistory를 참조하세요.

`DescribeAlarms`

다음 코드 예시는 `DescribeAlarms`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Describes the CloudWatch alarms of the 'METRIC_ALARM' type.
     *
     * @return a {@link CompletableFuture} that represents the asynchronous
operation
     * of describing the CloudWatch alarms. The future completes when the
     * operation is finished, either successfully or with an error.
     */
    public CompletableFuture<Void> describeAlarmsAsync() {
        List<AlarmType> typeList = new ArrayList<>();
        typeList.add(AlarmType.METRIC_ALARM);
        DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
            .alarmTypes(typeList)
            .maxRecords(10)
            .build();

        return getAsyncClient().describeAlarms(alarmsRequest)
```

```
            .thenAccept(response -> {
                List<MetricAlarm> alarmList = response.metricAlarms();
                for (MetricAlarm alarm : alarmList) {
                    logger.info("Alarm name: {}", alarm.alarmName());
                    logger.info("Alarm description: {} ", alarm.alarmDescription());
                }
            })
            .whenComplete((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to describe alarms: {}", ex.getMessage());
                } else {
                    logger.info("Successfully described alarms.");
                }
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeAlarms를 참조하세요.

`DescribeAlarmsForMetric`

다음 코드 예시는 `DescribeAlarmsForMetric`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Checks for a metric alarm in AWS CloudWatch.
     *
     * @param fileName the name of the file containing the JSON configuration for
the custom metric
     * @return a {@link CompletableFuture} that completes when the check for the
metric alarm is complete
     */
    public CompletableFuture<Void> checkForMetricAlarmAsync(String fileName) {
```

```
        CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
                return rootNode.toString(); // Return JSON as a string for further
processing
            } catch (IOException e) {
                throw new RuntimeException("Failed to read file", e);
            }
        });

        return readFileFuture.thenCompose(jsonContent -> {
            try {
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();

                DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

                return checkForAlarmAsync(metricRequest, customMetricName, 10);

            } catch (IOException e) {
                throw new RuntimeException("Failed to parse JSON content", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error checking metric alarm",
exception);
            }
        });
    }

    // Recursive method to check for the alarm.
```

```
   /**
    * Checks for the existence of an alarm asynchronously for the specified metric.
    *
    * @param metricRequest    the request to describe the alarms for the specified
metric
    * @param customMetricName the name of the custom metric to check for an alarm
    * @param retries          the number of retries to perform if no alarm is found
    * @return a {@link CompletableFuture} that completes when an alarm is found or
the maximum number of retries has been reached
    */
   private static CompletableFuture<Void>
checkForAlarmAsync(DescribeAlarmsForMetricRequest metricRequest, String
customMetricName, int retries) {
       if (retries == 0) {
           return CompletableFuture.completedFuture(null).thenRun(() ->
               logger.info("No Alarm state found for {} after 10 retries.",
customMetricName)
           );
       }

       return
(getAsyncClient().describeAlarmsForMetric(metricRequest).thenCompose(response -> {
           if (response.hasMetricAlarms()) {
               logger.info("Alarm state found for {}", customMetricName);
               return CompletableFuture.completedFuture(null); // Alarm found,
complete the future
           } else {
               return CompletableFuture.runAsync(() -> {
                   try {
                       Thread.sleep(20000);
                       logger.info(".");
                   } catch (InterruptedException e) {
                       throw new RuntimeException("Interrupted while waiting to
retry", e);
                   }
               }).thenCompose(v -> checkForAlarmAsync(metricRequest,
customMetricName, retries - 1)); // Recursive call
           }
       }));
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeAlarmsForMetric을 참조하세요.

`DescribeAnomalyDetectors`

다음 코드 예시는 `DescribeAnomalyDetectors`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Describes the anomaly detectors based on the specified JSON file.
    *
    * @param fileName the name of the JSON file containing the custom metric
namespace and name
    * @return a {@link CompletableFuture} that completes when the anomaly detectors
have been described
    * @throws RuntimeException if there is a failure during the operation, such as
when reading or parsing the JSON file,
    *                          or when describing the anomaly detectors
    */
   public CompletableFuture<Void> describeAnomalyDetectorsAsync(String fileName) {
       CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               return new ObjectMapper().readTree(parser);
           } catch (IOException e) {
               throw new RuntimeException("Failed to read or parse the file", e);
           }
       });

       return readFileFuture.thenCompose(rootNode -> {
           try {
               String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
               String customMetricName =
rootNode.findValue("customMetricName").asText();
```

```
                DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
                    .maxResults(10)
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

                return
getAsyncClient().describeAnomalyDetectors(detectorsRequest).thenAccept(response ->
{
                    List<AnomalyDetector> anomalyDetectorList =
response.anomalyDetectors();
                    for (AnomalyDetector detector : anomalyDetectorList) {
                        logger.info("Metric name: {} ",
detector.singleMetricAnomalyDetector().metricName());
                        logger.info("State: {} ", detector.stateValue());
                    }
                });
            } catch (RuntimeException e) {
                throw new RuntimeException("Failed to describe anomaly detectors",
e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error describing anomaly detectors",
exception);
            }
        });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeAnomalyDetectors를 참조하세요.

`DisableAlarmActions`

다음 코드 예시는 `DisableAlarmActions`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보에 대한 내용은 AWS SDK for Java 2.x API 참조의 DisableAlarmActions를 참조하세요.

`EnableAlarmActions`

다음 코드 예시는 `EnableAlarmActions`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 EnableAlarmActions를 참조하세요.

`GetMetricData`

다음 코드 예시는 `GetMetricData`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Retrieves custom metric data from the AWS CloudWatch service.
    *
    * @param fileName the name of the file containing the custom metric information
    * @return a {@link CompletableFuture} that completes when the metric data has
been retrieved
    */
   public CompletableFuture<Void> getCustomMetricDataAsync(String fileName) {
       CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
           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);
```

```
                return rootNode.toString(); // Return JSON as a string for further
processing
            } catch (IOException e) {
                throw new RuntimeException("Failed to read file", e);
            }
        });

        return readFileFuture.thenCompose(jsonContent -> {
            try {
                // Parse the JSON string to extract relevant values.
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();

                // Set the current time and date range for metric query.
                Instant nowDate = Instant.now();
                long hours = 1;
                long minutes = 30;
                Instant endTime = 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(60)  // Assuming period in seconds
                    .metric(met)
                    .build();

                MetricDataQuery dataQuery = MetricDataQuery.builder()
                    .metricStat(metStat)
                    .id("foo2")
                    .returnData(true)
                    .build();

                List<MetricDataQuery> dq = new ArrayList<>();
                dq.add(dataQuery);
```

```
                GetMetricDataRequest getMetricDataRequest =
GetMetricDataRequest.builder()
                    .maxDatapoints(10)
                    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
                    .startTime(nowDate)
                    .endTime(endTime)
                    .metricDataQueries(dq)
                    .build();

                // Call the async method for CloudWatch data retrieval.
                return getAsyncClient().getMetricData(getMetricDataRequest);

            } catch (IOException e) {
                throw new RuntimeException("Failed to parse JSON content", e);
            }
        }).thenAccept(response -> {
            List<MetricDataResult> data = response.metricDataResults();
            for (MetricDataResult item : data) {
                logger.info("The label is: {}", item.label());
                logger.info("The status code is: {}", item.statusCode().toString());
            }
        }).exceptionally(exception -> {
            throw new RuntimeException("Failed to get metric data", exception);
        });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetMetricData를 참조하세요.

`GetMetricStatistics`

다음 코드 예시는 `GetMetricStatistics`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
```

```
    * Retrieves and displays metric statistics for the specified parameters.
    *
    * @param nameSpace    the namespace for the metric
    * @param metVal       the name of the metric
    * @param metricOption the statistic to retrieve for the metric (e.g.,
"Maximum", "Average")
    * @param date         the date for which to retrieve the metric statistics, in
the format "yyyy-MM-dd'T'HH:mm:ss'Z'"
    * @param myDimension  the dimension(s) to filter the metric statistics by
    * @return a {@link CompletableFuture} that completes when the metric statistics
have been retrieved and displayed
    */
   public CompletableFuture<GetMetricStatisticsResponse>
getAndDisplayMetricStatisticsAsync(String nameSpace, String metVal,

        String metricOption, String date, Dimension myDimension) {

       Instant start = Instant.parse(date);
       Instant endDate = Instant.now();

       // Building the request for metric statistics.
       GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
           .endTime(endDate)
           .startTime(start)
           .dimensions(myDimension)
           .metricName(metVal)
           .namespace(nameSpace)
           .period(86400) // 1 day period
           .statistics(Statistic.fromValue(metricOption))
           .build();

       return getAsyncClient().getMetricStatistics(statisticsRequest)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   List<Datapoint> data = response.datapoints();
                   if (!data.isEmpty()) {
                       for (Datapoint datapoint : data) {
                           logger.info("Timestamp: {} Maximum value: {}",
datapoint.timestamp(), datapoint.maximum());
                       }
                   } else {
                       logger.info("The returned data list is empty");
                   }
```

```
                } else {
                    logger.info("Failed to get metric statistics: {} ",
exception.getMessage());
                }
            })
            .exceptionally(exception -> {
                throw new RuntimeException("Error while getting metric statistics: "
+ exception.getMessage(), exception);
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetMetricStatistics를 참조하세요.

`GetMetricWidgetImage`

다음 코드 예시는 `GetMetricWidgetImage`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Retrieves and saves a custom metric image to a file.
     *
     * @param fileName the name of the file to save the metric image to
     * @return a {@link CompletableFuture} that completes when the image has been
saved to the file
     */
    public CompletableFuture<Void> downloadAndSaveMetricImageAsync(String fileName)
{
        logger.info("Getting Image data for custom metric.");
        String myJSON = """
              {
                  "title": "Example Metric Graph",
                  "view": "timeSeries",
```

```
                "stacked ": false,
                "period": 10,
                "width": 1400,
                "height": 600,
                "metrics": [
                    [
                    "AWS/Billing",
                    "EstimatedCharges",
                    "Currency",
                    "USD"
                   ]
                ]
              }
            """;

       GetMetricWidgetImageRequest imageRequest =
GetMetricWidgetImageRequest.builder()
            .metricWidget(myJSON)
            .build();

       return getAsyncClient().getMetricWidgetImage(imageRequest)
            .thenCompose(response -> {
                SdkBytes sdkBytes = response.metricWidgetImage();
                byte[] bytes = sdkBytes.asByteArray();
                return CompletableFuture.runAsync(() -> {
                    try {
                        File outputFile = new File(fileName);
                        try (FileOutputStream outputStream = new
FileOutputStream(outputFile)) {
                            outputStream.write(bytes);
                        }
                    } catch (IOException e) {
                        throw new RuntimeException("Failed to write image to file",
e);
                    }
                });
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Error getting and saving metric
image", exception);
                } else {
                    logger.info("Image data saved successfully to {}", fileName);
                }
```

```
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetMetricWidgetImage를 참조하세요.

`ListDashboards`

다음 코드 예시는 `ListDashboards`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Lists the available dashboards.
     *
     * @return a {@link CompletableFuture} that completes when the operation is
finished.
     * The future will complete exceptionally if an error occurs while listing the
dashboards.
     */
    public CompletableFuture<Void> listDashboardsAsync() {
        ListDashboardsRequest listDashboardsRequest =
ListDashboardsRequest.builder().build();
        ListDashboardsPublisher paginator =
getAsyncClient().listDashboardsPaginator(listDashboardsRequest);
        return paginator.subscribe(response -> {
            response.dashboardEntries().forEach(entry -> {
                logger.info("Dashboard name is: {} ", entry.dashboardName());
                logger.info("Dashboard ARN is: {} ", entry.dashboardArn());
            });
        }).exceptionally(ex -> {
            logger.info("Failed to list dashboards: {} ", ex.getMessage());
            throw new RuntimeException("Error occurred while listing dashboards",
ex);
        });
```

```
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListDashboards를 참조하세요.

`ListMetrics`

다음 코드 예시는 `ListMetrics`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Retrieves a list of metric names for the specified namespace.
    *
    * @param namespace the namespace for which to retrieve the metric names
    * @return a {@link CompletableFuture} that, when completed, contains an {@link
ArrayList} of
    * the metric names in the specified namespace
    * @throws RuntimeException if an error occurs while listing the metrics
    */
   public CompletableFuture<ArrayList<String>> listMetsAsync(String namespace) {
       ListMetricsRequest request = ListMetricsRequest.builder()
           .namespace(namespace)
           .build();

       ListMetricsPublisher metricsPaginator =
getAsyncClient().listMetricsPaginator(request);
       Set<String> metSet = new HashSet<>();
       CompletableFuture<Void> future = metricsPaginator.subscribe(response -> {
           response.metrics().forEach(metric -> {
               String metricName = metric.metricName();
               metSet.add(metricName);
           });
       });

       return future
```

```
            .thenApply(ignored -> new ArrayList<>(metSet))
            .exceptionally(exception -> {
                throw new RuntimeException("Failed to list metrics: " +
exception.getMessage(), exception);
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListMetrics를 참조하세요.

`PutAnomalyDetector`

다음 코드 예시는 `PutAnomalyDetector`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Adds an anomaly detector for the given file.
     *
     * @param fileName the name of the file containing the anomaly detector
configuration
     * @return a {@link CompletableFuture} that completes when the anomaly detector
has been added
     */
    public CompletableFuture<Void> addAnomalyDetectorAsync(String fileName) {
        CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                return new ObjectMapper().readTree(parser); // Return the root node
            } catch (IOException e) {
                throw new RuntimeException("Failed to read or parse the file", e);
            }
        });
```

```
        return readFileFuture.thenCompose(rootNode -> {
            try {
                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();

                return
getAsyncClient().putAnomalyDetector(anomalyDetectorRequest).thenAccept(response ->
{
                    logger.info("Added anomaly detector for metric {}",
customMetricName);
                });
            } catch (Exception e) {
                throw new RuntimeException("Failed to create anomaly detector", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error adding anomaly detector",
exception);
            }
        });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutAnomalyDetector를 참조하세요.

PutDashboard

다음 코드 예시는 PutDashboard의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Creates a new dashboard with the specified name and metrics from the given
file.
    *
    * @param dashboardName the name of the dashboard to be created
    * @param fileName      the name of the file containing the dashboard body
    * @return a {@link CompletableFuture} representing the asynchronous operation
of creating the dashboard
    * @throws IOException if there is an error reading the dashboard body from the
file
    */
   public CompletableFuture<PutDashboardResponse>
createDashboardWithMetricsAsync(String dashboardName, String fileName) throws
IOException {
       String dashboardBody = readFileAsString(fileName);
       PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
           .dashboardName(dashboardName)
           .dashboardBody(dashboardBody)
           .build();

       return getAsyncClient().putDashboard(dashboardRequest)
           .handle((response, ex) -> {
               if (ex != null) {
                   logger.info("Failed to create dashboard: {}", ex.getMessage());
                   throw new RuntimeException("Dashboard creation failed", ex);
               } else {
                   // Handle the normal response case
                   logger.info("{} was successfully created.", dashboardName);
                   List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
                   if (messages.isEmpty()) {
                       logger.info("There are no messages in the new Dashboard.");
                   } else {
```

```
                    for (DashboardValidationMessage message : messages) {
                        logger.info("Message: {}", message.message());
                    }
                }
                return response; // Return the response for further use
            }
        });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutDashboard를 참조하세요.

`PutMetricAlarm`

다음 코드 예시는 `PutMetricAlarm`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Creates an alarm based on the configuration provided in a JSON file.
     *
     * @param fileName the name of the JSON file containing the alarm configuration
     * @return a CompletableFuture that represents the asynchronous operation of
creating the alarm
     * @throws RuntimeException if an exception occurs while reading the JSON file
or creating the alarm
     */
    public CompletableFuture<String> createAlarmAsync(String fileName) {
        com.fasterxml.jackson.databind.JsonNode rootNode;
        try {
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            rootNode = new ObjectMapper().readTree(parser);
        } catch (IOException e) {
            throw new RuntimeException("Failed to read the alarm configuration
file", e);
        }
```

```
        // Extract values from the JSON node.
        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();

        // Call the putMetricAlarm asynchronously and handle the result.
        return getAsyncClient().putMetricAlarm(alarmRequest)
            .handle((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to create alarm: {}", ex.getMessage());
                    throw new RuntimeException("Failed to create alarm", ex);
                } else {
                    logger.info("{} was successfully created!", alarmName);
                    return alarmName;
                }
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutMetricAlarm을 참조하세요.

`PutMetricData`

다음 코드 예시는 `PutMetricData`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Adds metric data for an alarm asynchronously.
    *
    * @param fileName the name of the JSON file containing the metric data
    * @return a CompletableFuture that asynchronously returns the
PutMetricDataResponse
    */
   public CompletableFuture<PutMetricDataResponse>
addMetricDataForAlarmAsync(String fileName) {
       CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
               return rootNode.toString(); // Return JSON as a string for further
processing
           } catch (IOException e) {
               throw new RuntimeException("Failed to read file", e);
           }
       });

       return readFileFuture.thenCompose(jsonContent -> {
           try {
```

```
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();
                Instant instant = Instant.now();

                // Create MetricDatum objects.
                MetricDatum datum1 = 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(datum1);
                metricDataList.add(datum2);

                // Build the PutMetricData request.
                PutMetricDataRequest request = PutMetricDataRequest.builder()
                    .namespace(customMetricNamespace)
                    .metricData(metricDataList)
                    .build();

                // Send the request asynchronously.
                return getAsyncClient().putMetricData(request);

            } catch (IOException e) {
                CompletableFuture<PutMetricDataResponse> failedFuture = new
CompletableFuture<>();
                failedFuture.completeExceptionally(new RuntimeException("Failed to
parse JSON content", e));
                return failedFuture;
            }
        }).whenComplete((response, exception) -> {
```

```
            if (exception != null) {
                logger.error("Failed to put metric data: " + exception.getMessage(),
exception);
            } else {
                logger.info("Added metric values for metric.");
            }
        });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutMetricData를 참조하세요.

시나리오

DynamoDB 성능 모니터링

다음 코드 예제는 성능 모니터링을 위해 애플리케이션의 DynamoDB 사용을 구성하는 방법을 보여줍니다.

SDK for Java 2.x

이 예제는 DynamoDB의 성능을 모니터링하도록 Java 애플리케이션을 구성하는 방법을 보여줍니다. 애플리케이션은 성능을 모니터링할 수 있는 CloudWatch로 지표 데이터를 전송합니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- CloudWatch
- DynamoDB

Java 2.x용 SDK를 사용하는 CloudWatch Events 예제

다음 코드 예제에서는 CloudWatch Events와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`PutEvents`

다음 코드 예시는 `PutEvents`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutEvents를 참조하세요.

`PutRule`

다음 코드 예시는 `PutRule`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutRule을 참조하세요.

PutTargets

다음 코드 예시는 PutTargets의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutTargets를 참조하세요.

Java 2.x용 SDK를 사용하는 CloudWatch Logs 예제

다음 코드 예제에서는 CloudWatch Logs와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`DeleteSubscriptionFilter`

다음 코드 예시는 `DeleteSubscriptionFilter`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
```

```
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteSubscriptionFilter 참조하세요.

`DescribeLogStreams`

다음 코드 예시는 `DescribeLogStreams`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

지정된 접두사와 일치하는 지정된 로그 그룹 내의 로그 스트림을 검색합니다.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CloudWatchLogsSearch {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <logGroupName> <logStreamName>

                Where:
                  logGroupName - The name of the log group (for example,
 WeathertopJavaContainerLogs).
                  logStreamName - The name of the log stream (for example,
 weathertop-java-stream).
```

```
                  pattern - the pattern to use (for example, INFO)

                """;

        if (args.length != 3) {
            System.out.print(usage);
            System.exit(1);
        }

        String logGroupName = args[0] ;
        String logStreamName = args[1] ;
        String pattern = args[2] ;

        CloudWatchLogsClient cwlClient = CloudWatchLogsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        searchLogStreamsAndFilterEvents(cwlClient, logGroupName, logStreamName,
pattern);
    }

    /**
     * Searches for log streams with a specific prefix within a log group and
filters log events based on a specified pattern.
     *
     * @param cwlClient       the CloudWatchLogsClient used to interact with AWS
CloudWatch Logs
     * @param logGroupName    the name of the log group to search within
     * @param logStreamPrefix the prefix of the log streams to search for
     * @param pattern         the pattern to filter log events by
     */
    public static void searchLogStreamsAndFilterEvents(CloudWatchLogsClient
cwlClient, String logGroupName, String logStreamPrefix, String pattern) {
        DescribeLogStreamsRequest describeLogStreamsRequest =
DescribeLogStreamsRequest.builder()
                .logGroupName(logGroupName)
                .logStreamNamePrefix(logStreamPrefix)
                .build();

        DescribeLogStreamsResponse describeLogStreamsResponse =
cwlClient.describeLogStreams(describeLogStreamsRequest);
        List<LogStream> logStreams = describeLogStreamsResponse.logStreams();

        for (LogStream logStream : logStreams) {
```

```
            String logStreamName = logStream.logStreamName();
            System.out.println("Searching in log stream: " + logStreamName);

            FilterLogEventsRequest filterLogEventsRequest =
 FilterLogEventsRequest.builder()
                    .logGroupName(logGroupName)
                    .logStreamNames(logStreamName)
                    .filterPattern(pattern)
                    .build();

            FilterLogEventsResponse filterLogEventsResponse =
 cwlClient.filterLogEvents(filterLogEventsRequest);

            for (FilteredLogEvent event : filterLogEventsResponse.events()) {
                System.out.println(event.message());
            }


 System.out.println("-------------------------------------------------"); //
 Separator for better readability
        }
    }
}
```

지정된 로그 그룹의 최신 로그 스트림에 대한 메타데이터를 출력합니다.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CloudWatchLogQuery {
    public static void main(final String[] args) {
        final String usage = """
                Usage:
                  <logGroupName>

                Where:
```

```
                  logGroupName - The name of the log group (for example, /aws/
lambda/ChatAIHandler).
                """;

        if (args.length != 1) {
            System.out.print(usage);
            System.exit(1);
        }

        String logGroupName = "/aws/lambda/ChatAIHandler" ; //args[0];
        CloudWatchLogsClient logsClient = CloudWatchLogsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        describeMostRecentLogStream(logsClient, logGroupName);
    }

    /**
     * Describes and prints metadata about the most recent log stream in the
 specified log group.
     *
     * @param logsClient   the CloudWatchLogsClient used to interact with AWS
 CloudWatch Logs
     * @param logGroupName the name of the log group
     */
    public static void describeMostRecentLogStream(CloudWatchLogsClient logsClient,
 String logGroupName) {
        DescribeLogStreamsRequest streamsRequest =
 DescribeLogStreamsRequest.builder()
                .logGroupName(logGroupName)
                .orderBy(OrderBy.LAST_EVENT_TIME)
                .descending(true)
                .limit(1)
                .build();

        try {
            DescribeLogStreamsResponse streamsResponse =
 logsClient.describeLogStreams(streamsRequest);
            List<LogStream> logStreams = streamsResponse.logStreams();

            if (logStreams.isEmpty()) {
                System.out.println("No log streams found for log group: " +
 logGroupName);
                return;
```

```
            }

            LogStream stream = logStreams.get(0);
            System.out.println("Most Recent Log Stream:");
            System.out.println("  Name: " + stream.logStreamName());
            System.out.println("  ARN: " + stream.arn());
            System.out.println("  Creation Time: " + stream.creationTime());
            System.out.println("  First Event Time: " +
 stream.firstEventTimestamp());
            System.out.println("  Last Event Time: " + stream.lastEventTimestamp());
            System.out.println("  Stored Bytes: " + stream.storedBytes());
            System.out.println("  Upload Sequence Token: " +
 stream.uploadSequenceToken());

        } catch (CloudWatchLogsException e) {
            System.err.println("Failed to describe log stream: " +
 e.awsErrorDetails().errorMessage());
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeLogStreams을 참조하세요.

`DescribeSubscriptionFilters`

다음 코드 예시는 `DescribeSubscriptionFilters`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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");
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeSubscriptionFilters 참조하세요.

`GetLogEvents`

다음 코드 예시는 `GetLogEvents`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DescribeLogStreamsRequest;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DescribeLogStreamsResponse;
import software.amazon.awssdk.services.cloudwatchlogs.model.GetLogEventsRequest;
import software.amazon.awssdk.services.cloudwatchlogs.model.GetLogEventsResponse;

import java.time.Instant;
import java.time.temporal.ChronoUnit;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetLogEvents {

    public static void main(String[] args) {

        final String usage = """

                Usage:
                  <logGroupName> <logStreamName>

                Where:
```

```
                    logGroupName - The name of the log group (for example,
 myloggroup).
                    logStreamName - The name of the log stream (for example,
 mystream).

                """;

        // if (args.length != 2) {
        //     System.out.print(usage);
        //     System.exit(1);
//        }

         String logGroupName = "WeathertopJavaContainerLogs" ; //args[0];
         String logStreamName = "weathertop-java-stream" ; //args[1];

         Region region = Region.US_EAST_1 ;
         CloudWatchLogsClient cloudWatchLogsClient = CloudWatchLogsClient.builder()
                 .region(region)
                 .build();

         getCWLogEvents(cloudWatchLogsClient, logGroupName, logStreamName);
         cloudWatchLogsClient.close();
    }

    public static void getCWLogEvents(CloudWatchLogsClient cloudWatchLogsClient,
                                      String logGroupName,
                                      String logStreamPrefix) {
        try {
            // First, find the exact log stream name
            DescribeLogStreamsRequest describeRequest =
 DescribeLogStreamsRequest.builder()
                    .logGroupName(logGroupName)
                    .logStreamNamePrefix(logStreamPrefix)
                    .limit(1) // get the first matching stream
                    .build();

            DescribeLogStreamsResponse describeResponse =
 cloudWatchLogsClient.describeLogStreams(describeRequest);

            if (describeResponse.logStreams().isEmpty()) {
                System.out.println("No matching log streams found for prefix: " +
 logStreamPrefix);
                return;
            }
```

```
            String exactLogStreamName =
 describeResponse.logStreams().get(0).logStreamName();
            System.out.println("Using exact log stream: " + exactLogStreamName);

            long startTime = Instant.now().minus(7, ChronoUnit.DAYS).toEpochMilli();
            long endTime = Instant.now().toEpochMilli();

            GetLogEventsRequest getLogEventsRequest = GetLogEventsRequest.builder()
                    .logGroupName(logGroupName)
                    .logStreamName(exactLogStreamName) // <-- exact name, not prefix
                    .startTime(startTime)
                    .endTime(endTime)
                    .startFromHead(true)
                    .build();

            GetLogEventsResponse response =
 cloudWatchLogsClient.getLogEvents(getLogEventsRequest);

            if (response.events().isEmpty()) {
                System.out.println("No log events found in the past 7 days.");
            } else {
                response.events().forEach(e -> System.out.println(e.message()));
            }

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetLogEvents를 참조하세요.

PutSubscriptionFilter

다음 코드 예시는 PutSubscriptionFilter의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutSubscriptionFilter 참조하세요.

`StartLiveTail`

다음 코드 예시는 `StartLiveTail`의 사용 방법을 보여줍니다.

SDK for Java 2.x

필수 파일을 포함합니다.

```
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;
```

Live Tail 세션의 이벤트를 처리합니다.

```
   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();
    }
```

Live Tail 세션을 시작합니다.

```
        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));
```

일정 시간이 경과하면 Live Tail 세션을 중단합니다.

```
        /* 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");
        }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartLiveTail을 참조하세요.

시나리오

예약된 이벤트를 사용하여 Lambda 함수 간접 호출

다음 코드 예제에서는 Amazon EventBridge 예약 이벤트에서 호출된 AWS Lambda 함수를 생성하는 방법을 보여줍니다.

SDK for Java 2.x

AWS Lambda 함수를 호출하는 Amazon EventBridge 예약 이벤트를 생성하는 방법을 보여줍니다. Lambda 함수가 간접 호출될 때 cron 표현식을 사용하여 일정을 예약하도록 EventBridge를 구성합니다. 이 예제에서는 Lambda Java 런타임 API를 사용하여 Lambda 함수를 생성합니다. 이 예제에서는 다양한 AWS 서비스를 호출하여 특정 사용 사례를 수행합니다. 이 예제에서는 1주년 기념일에 직원에게 축하하는 모바일 문자 메시지를 전송하는 앱을 생성하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

SDK for Java 2.x를 사용한 Amazon Cognito 자격 증명 예시

다음 코드 예제에서는 Amazon Cognito Identity와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`CreateIdentityPool`

다음 코드 예시는 `CreateIdentityPool`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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 "";
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateIdentityPool을 참조하세요.

`DeleteIdentityPool`

다음 코드 예시는 `DeleteIdentityPool`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteIdentityPool을 참조하세요.

`GetCredentialsForIdentity`

다음 코드 예시는 `GetCredentialsForIdentity`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetCredentialsForIdentity 섹션을 참조하세요.

`ListIdentityPools`

다음 코드 예시는 `ListIdentityPools`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는AWS SDK for Java 2.x API 참조의 ListIdentityPools를 참조하세요.

Java 2.x용 SDK를 사용하는 Amazon Cognito 자격 증명 공급자 예제

다음 코드 예제에서는 Amazon Cognito 자격 증명 공급자와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시작하기
- 작업
- 시나리오

시작하기

Hello Amazon Cognito

다음 코드 예시에서는 Amazon Cognito 사용을 시작하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListFunctions를 참조하세요.

작업

`AdminGetUser`

다음 코드 예시는 `AdminGetUser`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 AdminGetUser를 참조하세요.

`AdminInitiateAuth`

다음 코드 예시는 `AdminInitiateAuth`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 AdminInitiateAuth를 참조하세요.

`AdminRespondToAuthChallenge`

다음 코드 예시는 `AdminRespondToAuthChallenge`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    // 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());
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 AdminRespondToAuthChallenge를 참조하세요.

`AssociateSoftwareToken`

다음 코드 예시는 `AssociateSoftwareToken`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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();
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 AssociateSoftwareToken을 참조하세요.

`ConfirmSignUp`

다음 코드 예시는 `ConfirmSignUp`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ConfirmSignUp을 참조하세요.

`CreateUserPool`

다음 코드 예시는 `CreateUserPool`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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 "";
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateUserPool 섹션을 참조하세요.

`CreateUserPoolClient`

다음 코드 예시는 `CreateUserPoolClient`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateUserPoolClient 섹션을 참조하세요.

`ListUserPools`

다음 코드 예시는 `ListUserPools`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListFunctions를 참조하세요.

`ListUsers`

다음 코드 예시는 `ListUsers`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListUsers를 참조하세요.

`ResendConfirmationCode`

다음 코드 예시는 `ResendConfirmationCode`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ResendConfirmationCode를 참조하세요.

SignUp

다음 코드 예시는 SignUp의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 SignUp를 참조하세요.

`VerifySoftwareToken`

다음 코드 예시는 `VerifySoftwareToken`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 VerifySoftwareToken을 참조하세요.

시나리오

MFA가 필요한 사용자 풀에 사용자 가입시키기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 사용자 이름, 암호 및 이메일 주소로 사용자를 가입시키고 확인합니다.
- MFA 애플리케이션을 사용자와 연결하여 다중 인증을 설정합니다.
- 암호와 MFA 코드를 사용하여 로그인합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 항목을 참조하세요.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken

- ConfirmDevice
- ConfirmSignUp
- InitiateAuth
- ListUsers
- ResendConfirmationCode
- RespondToAuthChallenge
- SignUp
- VerifySoftwareToken

Java 2.x용 SDK를 사용하는 Amazon Comprehend 예제

다음 코드 예제에서는 AWS SDK for Java 2.x Amazon Comprehend에서를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업
- 시나리오

작업

`CreateDocumentClassifier`

다음 코드 예시는 `CreateDocumentClassifier`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateDocumentClassifier를 참조하세요.

`DetectDominantLanguage`

다음 코드 예시는 `DetectDominantLanguage`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DetectDominantLanguage를 참조하세요.

`DetectEntities`

다음 코드 예시는 `DetectEntities`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DetectEntities를 참조하세요.

`DetectKeyPhrases`

다음 코드 예시는 `DetectKeyPhrases`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DetectKeyPhrases를 참조하세요.

`DetectSentiment`

다음 코드 예시는 `DetectSentiment`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DetectSentiment를 참조하세요.

DetectSyntax

다음 코드 예시는 DetectSyntax의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DetectSyntax를 참조하세요.

시나리오

Amazon Lex 챗봇 구축

다음 코드 예제에서는 챗봇을 만들어 웹사이트 방문자를 참여시키는 방법을 보여줍니다.

SDK for Java 2.x

Amazon Lex API를 사용하여 웹 애플리케이션 내에 챗봇을 구축하여 웹 사이트 방문자의 참여를 유도하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

메시징 애플리케이션 생성

다음 코드 예제에서는 Amazon SQS를 사용하여 메시징 애플리케이션을 만드는 방법을 보여줍니다.

SDK for Java 2.x

Amazon SQS API를 사용하여 메시지를 보내고 검색하는 Spring REST API를 개발하는 방법을 보여 줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Comprehend
- Amazon SQS

고객 피드백 분석을 위한 애플리케이션 생성

다음 코드 예제에서는 고객 의견 카드를 분석하고, 원어에서 번역하고, 감정을 파악하고, 번역된 텍스트에서 오디오 파일을 생성하는 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

이 예제 애플리케이션은 고객 피드백 카드를 분석하고 저장합니다. 특히 뉴욕시에 있는 가상 호텔의 필요를 충족합니다. 호텔은 다양한 언어의 고객들로부터 물리적인 의견 카드의 형태로 피드백을 받습니다. 피드백은 웹 클라이언트를 통해 앱에 업로드됩니다. 의견 카드의 이미지가 업로드된 후 다음 단계가 수행됩니다.

- Amazon Textract를 사용하여 이미지에서 텍스트가 추출됩니다.
- Amazon Comprehend가 추출된 텍스트와 해당 언어의 감정을 파악합니다.
- 추출된 텍스트는 Amazon Translate를 사용하여 영어로 번역됩니다.
- Amazon Polly가 추출된 텍스트에서 오디오 파일을 합성합니다.

전체 앱은 AWS CDK를 사용하여 배포할 수 있습니다. 소스 코드와 배포 지침은 GitHub의 프로젝트를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Comprehend
- Lambda
- Amazon Polly

- Amazon Textract
- Amazon Translate

AWS Control Tower SDK for Java 2.x를 사용한 예제

다음 코드 예제에서는와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS Control Tower.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시작하기

안녕하세요 AWS Control Tower

다음 코드 예제에서는 AWS Control Tower를 사용하여 시작하는 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
public class HelloControlTower {
```

```
    public static void main(String[] args) {
        try {
            ControlTowerClient controlTowerClient = ControlTowerClient.builder()
                .build() ;
            helloControlTower(controlTowerClient);
        } catch (ControlTowerException e) {
            System.out.println("Control Tower error occurred: " +
e.awsErrorDetails().errorMessage());
        }
    }

    /**
     * Use the AWS SDK for Java (v2) to create an AWS Control Tower client
     * and list all available baselines.
     * This example uses the default settings specified in your shared credentials
     * and config files.
     *
     * @param controlTowerClient A ControlTowerClient object. This object wraps
     *                           the low-level AWS Control Tower service API.
     */
    public static void helloControlTower(ControlTowerClient controlTowerClient) {
        System.out.println("Hello, AWS Control Tower! Let's list available
baselines:\n");

        ListBaselinesIterable paginator = controlTowerClient.listBaselinesPaginator(
                ListBaselinesRequest.builder().build());
        List<String> baselineNames = new ArrayList<>();

        try {
            paginator.stream()
                .flatMap(response -> response.baselines().stream())
                    .forEach(baseline -> baselineNames.add(baseline.name()));

            System.out.println(baselineNames.size() + " baseline(s) retrieved.");
            for (String baselineName : baselineNames) {
                System.out.println("\t" + baselineName);
            }

        } catch (ControlTowerException e) {
            if ("AccessDeniedException".equals(e.awsErrorDetails().errorCode())) {
                System.out.println("Access denied. Please ensure you have the
necessary permissions.");
            } else {
                System.out.println("An error occurred: " + e.getMessage());
```

```
            }
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListBaselines을 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 랜딩 존을 나열합니다.
- 기준을 나열, 활성화, 조회, 재설정, 비활성화합니다.
- 제어 기능을 나열, 활성화, 조회, 비활성화합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

AWS Control Tower 기능을 보여주는 대화형 시나리오를 실행합니다.

```
public class ControlTowerScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final Scanner scanner = new Scanner(in);

    private static OrganizationsClient orgClient;
    private static ControlCatalogClient catClient;

    private static String ouId = null;
    private static String ouArn = null;
    private static String landingZoneArn = null;
    private static boolean useLandingZone = false;
```

```
    private String stack = null;
    private String accountId = null;

    public static void main(String[] args) {

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS Control Tower basics scenario!");
        System.out.println(DASHES);

        try {
            runScenarioAsync();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    // ------------------------------
    // Utilities
    // ------------------------------
    private static boolean askYesNo(String msg) {
        System.out.println(msg);
        return scanner.nextLine().trim().toLowerCase().startsWith("y");
    }

    private static void runScenarioAsync() {
        try {
            ControlTowerActions actions = new ControlTowerActions();

            // ------------------------------
            // Step 1: Landing Zones
            // ------------------------------
            System.out.println(DASHES);
            System.out.println("""
        Some demo operations require the use of a landing zone.
        You can use an existing landing zone or opt out of these operations in the
 demo.
        For instructions on how to set up a landing zone,
        see https://docs.aws.amazon.com/controltower/latest/userguide/getting-
started-from-console.html
        """);

            System.out.println("Step 1: Listing landing zones...");
            waitForInputToContinue(scanner);
```

```
            List<LandingZoneSummary> landingZones =
                    actions.listLandingZonesAsync().join();

            if (landingZones.isEmpty()) {
                System.out.println("No landing zones found. Landing-zone-dependent
steps will be skipped.");
                useLandingZone = false;
                waitForInputToContinue(scanner);
            } else {
                System.out.println("\nAvailable Landing Zones:");
                for (int i = 0; i < landingZones.size(); i++) {
                    System.out.printf("%d) %s%n", i + 1, landingZones.get(i).arn());
                }

                if (askYesNo("Do you want to use the first landing zone in the list
(" +
                        landingZones.get(0).arn() + ")? (y/n): ")) {
                    useLandingZone = true;
                    landingZoneArn = landingZones.get(0).arn();
                } else if (askYesNo("Do you want to use a different existing Landing
Zone for this demo? (y/n): ")) {
                    useLandingZone = true;
                    System.out.println("Enter landing zone ARN: ");
                    landingZoneArn = scanner.nextLine().trim();
                } else {
                    System.out.println("Proceeding without a landing zone.");
                    useLandingZone = false;
                    waitForInputToContinue(scanner);
                }
            }

            // ----------------------------
            // Setup Organization + Sandbox OU
            // ----------------------------
            if (useLandingZone) {
                System.out.println("Using landing zone ARN: " + landingZoneArn);

                ControlTowerActions.OrgSetupResult result =
                        actions.setupOrganizationAsync().join();

                ouArn = result.sandboxOuArn();
                ouId  = result.sandboxOuArn();

                System.out.println("Organization ID: " + result.orgId());
```

```
                System.out.println("Using Sandbox OU ARN: " + ouArn);
            }

            // ----------------------------
            // Step 2: Baselines
            // ----------------------------
            System.out.println(DASHES);
            System.out.println("Step 2: Listing available baselines...");
            System.out.println("""
In this step, the program lists available AWS Control Tower baselines and may
 perform
baseline-related operations (enable, disable, reset) if requested.

NOTE:
AWS Control Tower enforces governance through baselines and mandatory controls
(guardrails). Mandatory controls are required for landing zone governance and may
restrict certain operations depending on the account, region, or organizational
 policy.

For more information, see:
- Types of baselines in AWS Control Tower:
  https://docs.aws.amazon.com/controltower/latest/userguide/types-of-baselines.html
- Mandatory controls (guardrails) in AWS Control Tower:
  https://docs.aws.amazon.com/controltower/latest/controlreference/mandatory-
controls.html
- Baseline API examples:
  https://docs.aws.amazon.com/controltower/latest/userguide/baseline-api-
examples.html
""");



            waitForInputToContinue(scanner);
            List<BaselineSummary> baselines =
                    actions.listBaselinesAsync().join();

            BaselineSummary controlTowerBaseline = null;
            for (BaselineSummary b : baselines) {
                System.out.println("Baseline: " + b.name());
                System.out.println("  ARN: " + b.arn());
                if ("AWSControlTowerBaseline".equals(b.name())) {
                    controlTowerBaseline = b;
                }
            }
```

```
            waitForInputToContinue(scanner);

            if (useLandingZone && controlTowerBaseline != null) {

                System.out.println("\nListing enabled baselines:");
                List<EnabledBaselineSummary> enabledBaselines =
                        actions.listEnabledBaselinesAsync().join();

                String enabledBaselineArn = null;
                for (EnabledBaselineSummary eb : enabledBaselines) {
                    System.out.println("Checking enabled baseline ARN: " +
 eb.arn());
                    if (eb.baselineIdentifier().equals(controlTowerBaseline.arn()))
 {
                        enabledBaselineArn = eb.arn(); // correct enabled ARN for
 this baseline
                        break; // stop after finding the matching one
                    }
                }

                if (enabledBaselineArn == null) {
                    System.out.println("No enabled baseline found for " +
 controlTowerBaseline.arn());
                } else {
                    System.out.println("Selected enabled baseline ARN for reset/
disable: " + enabledBaselineArn);
                }


                // Enable the Baseline
                if (askYesNo("Do you want to enable the Control Tower Baseline? (y/
n): ")) {
                    System.out.println("\nEnabling Control Tower Baseline...");

                    String baselineId = controlTowerBaseline.arn();
                    String enabledBaselineId =
                            actions.enableBaselineAsync(
                                    ouArn,                  // targetIdentifier #
 the OU or account ARN
                                    baselineId,             // baselineIdentifier #
 the Control Tower baseline ARN
                                    "5.0"                   // baselineVersion #
 version string
```

```
                            ).join();


                    System.out.println("Enabled baseline operation ID: " +
enabledBaselineId);
                    if (enabledBaselineId == null) {
                       enabledBaselineId = enabledBaselineArn;
                    }

                    // Reset the Baseline
                    if (askYesNo("Do you want to reset the Control Tower Baseline?
(y/n): ")) {
                        String operationId =

actions.resetEnabledBaselineAsync(enabledBaselineId).join();
                        System.out.println("Reset baseline operation ID: " +
operationId);
                    }

                    if (askYesNo("Do you want to disable the Control Tower Baseline?
(y/n): ")) {
                        String operationId =

actions.disableBaselineAsync(enabledBaselineId).join();
                        System.out.println("Disabled baseline operation ID: " +
operationId);

                        System.out.println("Now we will re-enable the baseline and
wait 1 minute before making the call...");
                        try {
                            Thread.sleep(Duration.ofMinutes(1).toMillis());
                        } catch (InterruptedException e) {
                            Thread.currentThread().interrupt();
                            System.out.println("Wait interrupted");
                        }
                        String reEnabledBaselineId = actions.enableBaselineAsync(
                                ouArn,
                                baselineId,  // reuse baseline definition ARN
                                "5.0"
                        ).join();

                        System.out.println("Re-enabled baseline operation ID: " +
reEnabledBaselineId);
                    }
```

```
                }
            }

            // ----------------------------
            // Step 3: Controls
            // ----------------------------
            System.out.println(DASHES);
            System.out.println("Step 3: Managing Controls:");
            waitForInputToContinue(scanner);

            List<ControlSummary> controls =
                    actions.listControlsAsync().join();

            System.out.println("\nListing first 5 available Controls:");
            for (int i = 0; i < Math.min(5, controls.size()); i++) {
                ControlSummary c = controls.get(i);
                System.out.println("%d. %s - %s".formatted(i + 1, c.name(),
c.arn()));
            }

            if (useLandingZone) {
                waitForInputToContinue(scanner);

                List<EnabledControlSummary> enabledControls =
                        actions.listEnabledControlsAsync(ouArn).join();

                System.out.println("\nListing enabled controls:");
                for (int i = 0; i < enabledControls.size(); i++) {
                    System.out.println("%d. %s".formatted(i + 1,
enabledControls.get(i).controlIdentifier()));
                }

                String controlArnToEnable = null;
                for (ControlSummary control : controls) {
                    boolean enabled = enabledControls.stream()
                            .anyMatch(ec ->
ec.controlIdentifier().equals(control.arn()));
                    if (!enabled) {
                        controlArnToEnable = control.arn();
                        break;
                    }
                }

                waitForInputToContinue(scanner);
```

```
                if (controlArnToEnable != null &&
                        askYesNo("Do you want to enable the control " +
controlArnToEnable + "? (y/n): ")) {

                    String operationId =
                            actions.enableControlAsync(controlArnToEnable,
ouArn).join();

                    System.out.println("Enabled control with operation ID: " +
operationId);
                }

                waitForInputToContinue(scanner);

                if (controlArnToEnable != null &&
                        askYesNo("Do you want to disable the control? (y/n): ")) {

                    String operationId =
                            actions.disableControlAsync(controlArnToEnable,
ouArn).join();

                    System.out.println("Disable operation ID: " + operationId);
                }
            }
            System.out.println("\nThis concludes the example scenario.");
            System.out.println("Thanks for watching!");
            System.out.println(DASHES);

        } catch (CompletionException e) {
            Throwable cause = e.getCause() != null ? e.getCause() : e;
            System.out.println("Scenario failed: " + cause.getMessage());
            throw e; // bubble up for tests / callers
        } catch (Exception e) {
            System.out.println("Unexpected error running scenario: " +
e.getMessage());
            throw new RuntimeException(e);
        }
    }

    private static void waitForInputToContinue(Scanner sc) {
        System.out.println("\nEnter 'c' then <ENTER> to continue:");
        while (true) {
            String input = sc.nextLine();
            if ("c".equalsIgnoreCase(input.trim())) {
```

```
                System.out.println("Continuing...");
                break;
            }
        }
    }
}

public class ControlTowerActions {
    private static ControlCatalogAsyncClient controlCatalogAsyncClient;
    private static ControlTowerAsyncClient controlTowerAsyncClient;
    private static OrganizationsAsyncClient orgAsyncClient;

    private static OrganizationsAsyncClient getAsyncOrgClient() {
        if (orgAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                    .maxConcurrency(50)
                    .connectionTimeout(Duration.ofSeconds(60))
                    .readTimeout(Duration.ofSeconds(60))
                    .writeTimeout(Duration.ofSeconds(60))
                    .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
                    .build();

            orgAsyncClient = OrganizationsAsyncClient.builder()
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
        return orgAsyncClient;
    }

    private static ControlCatalogAsyncClient getAsyncCatClient() {
        if (controlCatalogAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                    .maxConcurrency(100)
                    .connectionTimeout(Duration.ofSeconds(60))
                    .readTimeout(Duration.ofSeconds(60))
                    .writeTimeout(Duration.ofSeconds(60))
                    .build();
```

```
            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
                    .retryStrategy(RetryMode.STANDARD)
                    .build();

            controlCatalogAsyncClient = ControlCatalogAsyncClient.builder()
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
        return controlCatalogAsyncClient;
    }

    private static ControlTowerAsyncClient getAsyncClient() {
        if (controlTowerAsyncClient == null) {

            SdkAsyncHttpClient httpClient =
                    AwsCrtAsyncHttpClient.builder()
                            .maxConcurrency(100)
                            .connectionTimeout(Duration.ofSeconds(60))
                            .build();

            ClientOverrideConfiguration overrideConfig =
                    ClientOverrideConfiguration.builder()
                            .apiCallTimeout(Duration.ofMinutes(2))
                            .apiCallAttemptTimeout(Duration.ofSeconds(90))
                            .retryStrategy(RetryMode.STANDARD)
                            .build();

            controlTowerAsyncClient =
                    ControlTowerAsyncClient.builder()
                            .httpClient(httpClient)
                            .overrideConfiguration(overrideConfig)
                            .build();
        }

        return controlTowerAsyncClient;
    }

    public record OrgSetupResult(String orgId, String sandboxOuArn) {
    }
```

```
    public CompletableFuture<OrgSetupResult> setupOrganizationAsync() {
        System.out.println("Starting organization setup…");

        OrganizationsAsyncClient client = getAsyncOrgClient();

        // Step 1: Describe or create organization
        CompletableFuture<Organization> orgFuture = client.describeOrganization()
                .thenApply(desc -> {
                    System.out.println("Organization exists: " +
desc.organization().id());
                    return desc.organization();
                })
                .exceptionallyCompose(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;
                    if (cause instanceof AwsServiceException awsEx &&

"AWSOrganizationsNotInUseException".equals(awsEx.awsErrorDetails().errorCode())) {
                        System.out.println("No organization found. Creating one…");
                        return
client.createOrganization(CreateOrganizationRequest.builder()
                                        .featureSet(OrganizationFeatureSet.ALL)
                                        .build())
                                .thenApply(createResp -> {
                                    System.out.println("Created organization: {}" +
createResp.organization().id());
                                    return createResp.organization();
                                });
                    }
                    return CompletableFuture.failedFuture(
                            new CompletionException("Failed to describe or create
organization", cause)
                    );
                });

        // Step 2: Locate Sandbox OU
        return orgFuture.thenCompose(org -> {
            String orgId = org.id();
            System.out.println("Organization ID: {}" + orgId);

            return client.listRoots()
                    .thenCompose(rootsResp -> {
                        if (rootsResp.roots().isEmpty()) {
                            return CompletableFuture.failedFuture(
```

```
                                    new RuntimeException("No root found in
organization")
                            );
                        }
                        String rootId = rootsResp.roots().get(0).id();

                        ListOrganizationalUnitsForParentRequest ouRequest =
                                ListOrganizationalUnitsForParentRequest.builder()
                                        .parentId(rootId)
                                        .build();

                        ListOrganizationalUnitsForParentPublisher paginator =

client.listOrganizationalUnitsForParentPaginator(ouRequest);

                        AtomicReference<String> sandboxOuArnRef = new
AtomicReference<>();
                        return paginator.subscribe(page -> {
                                    for (OrganizationalUnit ou :
page.organizationalUnits()) {
                                        if ("Sandbox".equals(ou.name())) {
                                            sandboxOuArnRef.set(ou.arn());
                                            System.out.println("Found Sandbox OU: "
+ ou.id());

                                            break;
                                        }
                                    }
                                })
                                .thenApply(v -> {
                                    String sandboxArn = sandboxOuArnRef.get();
                                    if (sandboxArn == null) {
                                        System.out.println("Sandbox OU not found.");
                                    }
                                    return new OrgSetupResult(orgId, sandboxArn);
                                });
                    });
        }).exceptionally(ex -> {
            Throwable cause = ex.getCause() != null ? ex.getCause() : ex;
            System.out.println("Failed to setup organization: {}" +
cause.getMessage());
            throw new CompletionException(cause);
        });
    }
```

```
    /**
     * Lists all landing zones using pagination to retrieve complete results.
     *
     * @return a list of all landing zones
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<LandingZoneSummary>> listLandingZonesAsync() {
        System.out.println("Starting list landing zones paginator…");

        ListLandingZonesRequest request = ListLandingZonesRequest.builder().build();
        ListLandingZonesPublisher paginator =
 getAsyncClient().listLandingZonesPaginator(request);
        List<LandingZoneSummary> landingZones = new ArrayList<>();

        return paginator.subscribe(response -> {
                    if (response.landingZones() != null && !
response.landingZones().isEmpty()) {
                        response.landingZones().forEach(lz -> {
                            System.out.println("Landing zone ARN: " + lz.arn());
                            landingZones.add(lz);
                        });
                    } else {
                        System.out.println("Page contained no landing zones.");
                    }
                })
                .thenRun(() -> System.out.println("Successfully retrieved "+
 landingZones.size() + " landing zones." ))
                .thenApply(v -> landingZones)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();
                        switch (errorCode) {
                            case "AccessDeniedException":
                                throw new CompletionException(
                                        "Access denied when listing landing zones: "
 + e.getMessage(), e);
                            default:
                                throw new CompletionException(
                                        "Error listing landing zones: " +
 e.getMessage(), e);
                        }
```

```
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing landing zones: " +
 cause.getMessage(), cause);
                    }

                    throw new CompletionException("Failed to list landing zones",
 cause);
                });
    }


    /**
     * Lists all available baselines using pagination to retrieve complete results.
     *
     * @return a list of all baselines
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<BaselineSummary>> listBaselinesAsync() {
        System.out.println("Starting list baselines paginator…");
        ListBaselinesRequest request = ListBaselinesRequest.builder().build();
        ListBaselinesPublisher paginator =
                getAsyncClient().listBaselinesPaginator(request);

        List<BaselineSummary> baselines = new ArrayList<>();
        return paginator.subscribe(response -> {
                    if (response.baselines() != null && !
response.baselines().isEmpty()) {
                        response.baselines().forEach(baseline -> {
                            baselines.add(baseline);
                        });
                    } else {
                        System.out.println("Page contained no baselines.");
                    }
                })
                .thenRun(() ->
                        System.out.println("Successfully listed baselines. Total: "
 + baselines.size())
                )
                .thenApply(v -> baselines)
                .exceptionally(ex -> {
```

```
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("AccessDeniedException".equals(errorCode)) {
                            throw new CompletionException(
                                    "Access denied when listing baselines:
%s".formatted(e.getMessage()),
                                    e
                            );
                        }

                        throw new CompletionException(
                                "Error listing baselines:
%s".formatted(e.getMessage()),
                                e
                        );
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing baselines:
%s".formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException("Failed to list baselines",
cause);
                });
    }

    /**
     * Lists all enabled baselines using pagination to retrieve complete results.
     *
     * @return a list of all enabled baselines
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<EnabledBaselineSummary>>
listEnabledBaselinesAsync() {
        System.out.println("Starting list enabled baselines paginator…");
```

```
        ListEnabledBaselinesRequest request =
                ListEnabledBaselinesRequest.builder().build();

        ListEnabledBaselinesPublisher paginator =
                getAsyncClient().listEnabledBaselinesPaginator(request);

        List<EnabledBaselineSummary> enabledBaselines = new ArrayList<>();
        return paginator.subscribe(response -> {
                    if (response.enabledBaselines() != null
                            && !response.enabledBaselines().isEmpty()) {

                        response.enabledBaselines().forEach(baseline -> {
                            enabledBaselines.add(baseline);
                        });
                    } else {
                        System.out.println("Page contained no enabled baselines.");
                    }
                })
                .thenRun(() ->
                        System.out.println(
                                "Successfully listed enabled baselines. Total: " +
                                        enabledBaselines.size()
                        )
                )
                .thenApply(v -> enabledBaselines)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("AccessDeniedException".equals(errorCode)) {
                            throw new CompletionException(
                                    "Access denied when listing enabled baselines:
%s".formatted(e.getMessage()), e);
                        }

                        throw new CompletionException(
                                "Error listing enabled baselines: %s"
                                        .formatted(e.getMessage()),
                                e
                        );
                    }
```

```
                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing enabled baselines: %s"
                                        .formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException(
                            "Failed to list enabled baselines",
                            cause
                    );
                });
    }


    /**
     * Asynchronously enables a baseline for the specified target if not already
enabled.
     *
     * @param targetIdentifier       The ARN of the target (OU or account).
     * @param baselineIdentifier     The baseline definition ARN to enable.
     * @param baselineVersion        The baseline version to enable.
     * @return A CompletableFuture containing the enabled baseline ARN, or null if
already enabled.
     */
    public CompletableFuture<String> enableBaselineAsync(
            String targetIdentifier,
            String baselineIdentifier,
            String baselineVersion
    ) {
        EnableBaselineRequest request = EnableBaselineRequest.builder()
                .baselineIdentifier(baselineIdentifier)
                .baselineVersion(baselineVersion)
                .targetIdentifier(targetIdentifier)
                .build();

        return getAsyncClient().enableBaseline(request)
                .handle((resp, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ControlTowerException e) {
```

```
                            String code = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                            String msg = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();

                            if ("ValidationException".equals(code) &&
msg.contains("already enabled")) {
                                System.out.println("Baseline is already enabled for
this target # fetching ARN...");
                                return fetchEnabledBaselineArn(targetIdentifier,
baselineIdentifier)
                                        .join(); // fetch existing ARN synchronously
                            }

                            throw new RuntimeException("Error enabling baseline: " +
code + " - " + msg, e);
                        }

                        throw new RuntimeException("Unexpected error enabling
baseline: " + cause.getMessage(), cause);
                    }

                    return resp;
                })
                .thenCompose(result -> {
                    if (result instanceof EnableBaselineResponse resp) {
                        String operationId = resp.operationIdentifier();
                        String enabledBaselineArn = resp.arn();
                        System.out.println("Baseline enable started. ARN: " +
enabledBaselineArn
                                + ", operation ID: " + operationId);

                        // Inline polling
                        return CompletableFuture.supplyAsync(() -> {
                            while (true) {
                                GetBaselineOperationRequest opReq =
GetBaselineOperationRequest.builder()
                                        .operationIdentifier(operationId)
                                        .build();

                                GetBaselineOperationResponse opResp =
getAsyncClient().getBaselineOperation(opReq).join();
                                BaselineOperation op = opResp.baselineOperation();
                                BaselineOperationStatus status = op.status();
```

```
                                System.out.println("Operation " + operationId + "
status: " + status);

                                if (status == BaselineOperationStatus.SUCCEEDED) {
                                    return enabledBaselineArn;
                                } else if (status == BaselineOperationStatus.FAILED)
{
                                    String opId = op.operationIdentifier();
                                    String reason = op.statusMessage() != null ?
op.statusMessage() : "No failure reason provided";
                                    throw new RuntimeException("Baseline operation
failed (ID: " + opId + "), status: "
                                            + status + ", reason: " + reason);
                                }

                                try {
                                    Thread.sleep(Duration.ofSeconds(15).toMillis());
                                } catch (InterruptedException e) {
                                    Thread.currentThread().interrupt();
                                    throw new RuntimeException(e);
                                }
                            }
                        });
                } else if (result instanceof String existingArn) {
                    // Already enabled branch
                    return CompletableFuture.completedFuture(existingArn);
                }

                return CompletableFuture.completedFuture(null);
            });
    }


    /**
     * Fetches the ARN of an already-enabled baseline for the target asynchronously.
     */
    private CompletableFuture<String> fetchEnabledBaselineArn(String
targetIdentifier, String baselineIdentifier) {
        return
getAsyncClient().listEnabledBaselines(ListEnabledBaselinesRequest.builder().build())
                .thenApply(listResp -> {
                    for (EnabledBaselineSummary eb : listResp.enabledBaselines()) {
                        if (baselineIdentifier.equals(eb.baselineIdentifier())
                                && targetIdentifier.equals(eb.targetIdentifier())) {
```

```
                        return eb.arn();
                    }
                }
                return null; // not yet available
            });
    }


    /**
     * Disables a baseline for a specified target.
     *
     * @param enabledBaselineIdentifier the identifier of the enabled baseline to
disable
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> disableBaselineAsync(String
enabledBaselineIdentifier) {

        System.out.println("Starting disable of enabled baseline…");
        System.out.println("This operation will check the status every 15 seconds
until it completes (SUCCEEDED or FAILED).");

        DisableBaselineRequest request = DisableBaselineRequest.builder()
                .enabledBaselineIdentifier(enabledBaselineIdentifier)
                .build();

        return getAsyncClient().disableBaseline(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Disable baseline operation ID: " +
operationId);

                    // CompletableFuture that will be completed when operation
finishes
                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    // Polling loop
                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getBaselineOperationAsync(operationId)
```

```
                                    .thenAccept(statusObj -> {
                                        String status = statusObj.toString(); //
Convert enum/status to string for printing
                                        System.out.println("Current disable
operation status: " + status + " # waiting for SUCCEEDED or FAILED...");

                                        if ("SUCCEEDED".equalsIgnoreCase(status) ||
"FAILED".equalsIgnoreCase(status)) {
                                            System.out.println("Disable operation
finished with status: " + status);
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Schedule next poll in 15 seconds
                                            CompletableFuture.delayedExecutor(15,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        System.out.println("Error checking baseline
operation status: " + ex.getMessage());
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start first poll immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                        String errorMessage = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();

                        System.out.println("ControlTowerException caught while
disabling baseline: Code=" + errorCode + ", Message=" + errorMessage);
                        return null;
```

```
                    }

                    if (cause instanceof SdkException sdkEx) {
                        System.out.println("SDK exception caught while disabling
baseline: " + sdkEx.getMessage());
                        return null;
                    }

                    System.out.println("Unexpected exception while disabling
baseline: " + cause.getMessage());
                    return null;
                });
    }



    /**
     * Gets the status of a baseline operation.
     *
     * @param operationIdentifier the identifier of the operation
     * @return the operation status
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<BaselineOperationStatus> getBaselineOperationAsync(
            String operationIdentifier) {

        GetBaselineOperationRequest request = GetBaselineOperationRequest.builder()
                .operationIdentifier(operationIdentifier)
                .build();

        return getAsyncClient().getBaselineOperation(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null
                                ? exception.getCause()
                                : exception;

                        if (cause instanceof ControlTowerException e) {
                            String errorCode = e.awsErrorDetails().errorCode();

                            if ("ResourceNotFoundException".equals(errorCode)) {
                                throw new CompletionException(
                                        "Baseline operation not found: %s"
```

```
                                            .formatted(e.getMessage()),
                                        e
                                    );
                            }

                            throw new CompletionException(
                                    "Error getting baseline operation status: %s"
                                            .formatted(e.getMessage()),
                                    e
                            );
                        }

                        if (cause instanceof SdkException) {
                            throw new CompletionException(
                                    "SDK error getting baseline operation status:
%s"
                                            .formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to get baseline operation status",
                                cause
                        );
                    }
                })
                .thenApply(response -> {
                    BaselineOperationStatus status =
                            response.baselineOperation().status();
                    return status;
                });
    }


    /**
     * Lists all enabled controls for a specific target using pagination.
     *
     * @param targetIdentifier the identifier of the target (e.g., OU ARN)
     * @return a list of enabled controls
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
```

```
    public CompletableFuture<List<EnabledControlSummary>>
 listEnabledControlsAsync(String targetIdentifier) {
        System.out.println("Starting list enabled controls paginator for target " +
 targetIdentifier);
        ListEnabledControlsRequest request = ListEnabledControlsRequest.builder()
                .targetIdentifier(targetIdentifier)
                .build();

        ListEnabledControlsPublisher paginator =
 getAsyncClient().listEnabledControlsPaginator(request);
        List<EnabledControlSummary> enabledControls = new ArrayList<>();

        // Subscribe to the paginator asynchronously
        return paginator.subscribe(response -> {
                    if (response.enabledControls() != null && !
response.enabledControls().isEmpty()) {
                        response.enabledControls().forEach(control -> {
                            enabledControls.add(control);
                        });
                    } else {
                        System.out.println("Page contained no enabled controls.");
                    }
                })
                .thenRun(() -> System.out.println(
                        "Successfully retrieved "+enabledControls.size() +" enabled
 controls for target "+targetIdentifier
                ))
                .thenApply(v -> enabledControls)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        switch (errorCode) {
                            case "AccessDeniedException":
                                throw new CompletionException(
                                        "Access denied when listing enabled
 controls: %s".formatted(e.getMessage()), e);

                            case "ResourceNotFoundException":
                                if (e.getMessage() != null &&
 e.getMessage().contains("not registered with AWS Control Tower")) {
                                    throw new CompletionException(
```

```
                                                "Control Tower must be enabled to work
with controls", e);
                                }
                                throw new CompletionException(
                                        "Target not found when listing enabled
controls: %s".formatted(e.getMessage()), e);

                            default:
                                throw new CompletionException(
                                        "Error listing enabled controls:
%s".formatted(e.getMessage()), e);
                            }
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing enabled controls:
%s".formatted(cause.getMessage()), cause);
                    }

                    throw new CompletionException("Failed to list enabled controls",
cause);
                });
    }


    /**
     * Enables a control for a specified target.
     *
     * @param controlIdentifier the identifier of the control to enable
     * @param targetIdentifier  the identifier of the target (e.g., OU ARN)
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> enableControlAsync(
            String controlIdentifier,
            String targetIdentifier) {

        EnableControlRequest request = EnableControlRequest.builder()
                .controlIdentifier(controlIdentifier)
                .targetIdentifier(targetIdentifier)
                .build();
```

```
        return getAsyncClient().enableControl(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Enable control operation started. Operation
ID: " + operationId);

                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getControlOperationAsync(operationId)
                                    .thenAccept(status -> {
                                        System.out.println("Control operation
status: " + status);

                                        if (status ==
ControlOperationStatus.SUCCEEDED
                                                || status ==
ControlOperationStatus.FAILED) {
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Poll again after 30 seconds
                                            CompletableFuture.delayedExecutor(30,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start polling immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;
```

```
                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();
                        String message = e.getMessage() != null ? e.getMessage() :
"";

                        if ("ValidationException".equals(errorCode)
                                && message.contains("already enabled")) {
                            System.out.println("Control is already enabled for this
target");
                            return null;
                        }

                        if ("ResourceNotFoundException".equals(errorCode)
                                && message.contains("not registered with AWS Control
Tower")) {
                            System.out.println(
                                    "Control Tower must be enabled to work with
controls.");
                            return null;
                        }

                        throw new CompletionException(
                                "Couldn't enable control: %s".formatted(message),
                                e
                        );
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error enabling control: %s"
                                        .formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException(
                            "Failed to enable control",
                            cause
                    );
                });
    }


    /**
```

```
     * Disables a control for a specified target.
     *
     * @param controlIdentifier the identifier of the control to disable
     * @param targetIdentifier  the identifier of the target (e.g., OU ARN)
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> disableControlAsync(
            String controlIdentifier,
            String targetIdentifier) {

        DisableControlRequest request = DisableControlRequest.builder()
                .controlIdentifier(controlIdentifier)
                .targetIdentifier(targetIdentifier)
                .build();

        return getAsyncClient().disableControl(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Disable control operation started. Operation
ID: " + operationId);

                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getControlOperationAsync(operationId)
                                    .thenAccept(status -> {
                                        System.out.println("Control operation
status: " + status);

                                        if (status ==
ControlOperationStatus.SUCCEEDED
                                                || status ==
ControlOperationStatus.FAILED) {
                                            resultFuture.complete(operationId);
                                        } else {
                                            // poll again after 30 seconds
                                            CompletableFuture.delayedExecutor(30,
TimeUnit.SECONDS)
                                                    .execute(this);
```

```
                                    }
                                })
                                .exceptionally(ex -> {
                                    resultFuture.completeExceptionally(ex);
                                    return null;
                                });
                    }
                };

                // start polling immediately
                poller.run();

                return resultFuture;
            })
            .exceptionally(ex -> {
                Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                if (cause instanceof ControlTowerException e) {
                    String errorCode = e.awsErrorDetails().errorCode();

                    if ("ResourceNotFoundException".equals(errorCode)) {
                        // SPEC: notify user and continue
                        System.out.println("Control not found for disabling: " +
e.getMessage());
                        return null;
                    }

                    throw new CompletionException(
                            "Error disabling control: " + e.getMessage(), e);
                }

                if (cause instanceof SdkException) {
                    throw new CompletionException(
                            "SDK error disabling control: " +
cause.getMessage(), cause);
                }

                throw new CompletionException(
                        "Failed to disable control", cause);
            });
    }
```

```
    /**
     * Gets the status of a control operation.
     *
     * @param operationIdentifier the identifier of the operation
     * @return the operation status
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<ControlOperationStatus> getControlOperationAsync(
            String operationIdentifier) {

        GetControlOperationRequest request = GetControlOperationRequest.builder()
                .operationIdentifier(operationIdentifier)
                .build();

        return getAsyncClient().getControlOperation(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ControlTowerException e) {
                            String errorCode = e.awsErrorDetails().errorCode();

                            if ("ResourceNotFoundException".equals(errorCode)) {
                                throw new CompletionException(
                                        "Control operation not found:
%s".formatted(e.getMessage()),
                                        e
                                );
                            }

                            throw new CompletionException(
                                    "Error getting control operation status:
%s".formatted(e.getMessage()),
                                    e
                            );
                        }

                        if (cause instanceof SdkException) {
                            throw new CompletionException(
                                    "SDK error getting control operation status:
%s".formatted(cause.getMessage()),
                                    cause
```

```
                            );
                        }

                        throw new CompletionException("Failed to get control
 operation status", cause);
                    }
                })
                .thenApply(response -> response.controlOperation().status());
    }


    /**
     * Lists all controls in the Control Tower control catalog.
     *
     * @return a list of controls
     * @throws SdkException if a service-specific error occurs
     */
    public CompletableFuture<List<ControlSummary>> listControlsAsync() {
        System.out.println("Starting list controls paginator…");

        ListControlsRequest request = ListControlsRequest.builder().build();
        ListControlsPublisher paginator =
 getAsyncCatClient().listControlsPaginator(request);
        List<ControlSummary> controls = new ArrayList<>();

        return paginator.subscribe(response -> {
                    if (response.controls() != null && !
response.controls().isEmpty()) {
                        response.controls().forEach(control -> {
                            controls.add(control);
                        });
                    } else {
                        System.out.println("Page contained no controls.");
                    }
                })
                .thenRun(() -> System.out.println("Successfully retrieved " +
 controls.size() +" controls."))
                .thenApply(v -> controls)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof SdkException sdkEx) {
                        if (sdkEx.getMessage() != null &&
 sdkEx.getMessage().contains("AccessDeniedException")) {
```

```
                            throw new CompletionException(
                                    "Access denied when listing controls. Please
ensure you have the necessary permissions.",
                                    sdkEx
                            );
                        } else {
                            throw new CompletionException(
                                    "SDK error listing controls:
%s".formatted(sdkEx.getMessage()),
                                    sdkEx
                            );
                        }
                    }

                    throw new CompletionException("Failed to list controls", cause);
                });
    }

    /**
     * Resets an enabled baseline for a specific target.
     *
     * @param enabledBaselineIdentifier the identifier of the enabled baseline to
reset
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> resetEnabledBaselineAsync(String
enabledBaselineIdentifier) {

        System.out.println("Starting reset of enabled baseline…");
        System.out.println("This operation will check the status every 15 seconds
until it completes (SUCCEEDED or FAILED).");

        ResetEnabledBaselineRequest request = ResetEnabledBaselineRequest.builder()
                .enabledBaselineIdentifier(enabledBaselineIdentifier)
                .build();

        return getAsyncClient().resetEnabledBaseline(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Reset enabled baseline operation ID: " +
operationId);
```

```
                    // Polling loop
                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getBaselineOperationAsync(operationId)
                                    .thenAccept(statusObj -> {
                                        String status = statusObj.toString(); //
Convert enum/status to string for printing
                                        System.out.println("Current baseline
operation status: " + status + " # waiting for SUCCEEDED or FAILED...");

                                        if ("SUCCEEDED".equalsIgnoreCase(status) ||
"FAILED".equalsIgnoreCase(status)) {
                                            System.out.println("Baseline operation
finished with status: " + status);
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Schedule next poll in 15 seconds
                                            CompletableFuture.delayedExecutor(15,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        System.out.println("Error checking baseline
operation status: " + ex.getMessage());
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start first poll immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
```

```
                        String errorCode = e.awsErrorDetails() != null ?
 e.awsErrorDetails().errorCode() : "UNKNOWN";
                        String errorMessage = e.awsErrorDetails() != null ?
 e.awsErrorDetails().errorMessage() : e.getMessage();

                        System.out.println("ControlTowerException caught: Code=" +
 errorCode + ", Message=" + errorMessage);
                        return null;
                    }

                    if (cause instanceof SdkException sdkEx) {
                        System.out.println("SDK exception caught: " +
 sdkEx.getMessage());
                        return null;
                    }

                    System.out.println("Unexpected exception resetting baseline: " +
 cause.getMessage());
                    return null;
                });
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - CreateLandingZone
  - DeleteLandingZone
  - DisableBaseline
  - DisableControl
  - EnableBaseline
  - EnableControl
  - GetControlOperation
  - GetLandingZoneOperation
  - ListBaselines
  - ListEnabledBaselines
  - ListEnabledControls
  - ListLandingZones
  - ResetEnabledBaseline

작업

`DisableBaseline`

다음 코드 예시는 `DisableBaseline`의 사용 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Disables a baseline for a specified target.
    *
    * @param enabledBaselineIdentifier the identifier of the enabled baseline to
disable
    * @return the operation identifier
    * @throws ControlTowerException if a service-specific error occurs
    * @throws SdkException          if an SDK error occurs
    */
   public CompletableFuture<String> disableBaselineAsync(String
enabledBaselineIdentifier) {

       System.out.println("Starting disable of enabled baseline…");
       System.out.println("This operation will check the status every 15 seconds
until it completes (SUCCEEDED or FAILED).");

       DisableBaselineRequest request = DisableBaselineRequest.builder()
               .enabledBaselineIdentifier(enabledBaselineIdentifier)
               .build();

       return getAsyncClient().disableBaseline(request)
               .thenCompose(response -> {
                   String operationId = response.operationIdentifier();
                   System.out.println("Disable baseline operation ID: " +
operationId);
```

```
                    // CompletableFuture that will be completed when operation
finishes
                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    // Polling loop
                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getBaselineOperationAsync(operationId)
                                    .thenAccept(statusObj -> {
                                        String status = statusObj.toString(); //
Convert enum/status to string for printing
                                        System.out.println("Current disable
operation status: " + status + " # waiting for SUCCEEDED or FAILED...");

                                        if ("SUCCEEDED".equalsIgnoreCase(status) ||
"FAILED".equalsIgnoreCase(status)) {
                                            System.out.println("Disable operation
finished with status: " + status);
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Schedule next poll in 15 seconds
                                            CompletableFuture.delayedExecutor(15,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        System.out.println("Error checking baseline
operation status: " + ex.getMessage());
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start first poll immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;
```

```
                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                        String errorMessage = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();

                        System.out.println("ControlTowerException caught while
disabling baseline: Code=" + errorCode + ", Message=" + errorMessage);
                        return null;
                    }

                    if (cause instanceof SdkException sdkEx) {
                        System.out.println("SDK exception caught while disabling
baseline: " + sdkEx.getMessage());
                        return null;
                    }

                    System.out.println("Unexpected exception while disabling
baseline: " + cause.getMessage());
                    return null;
                });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DisableBaseline을 참조하세요.

`DisableControl`

다음 코드 예시는 `DisableControl`의 사용 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
```

```
     * Disables a control for a specified target.
     *
     * @param controlIdentifier the identifier of the control to disable
     * @param targetIdentifier  the identifier of the target (e.g., OU ARN)
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> disableControlAsync(
            String controlIdentifier,
            String targetIdentifier) {

        DisableControlRequest request = DisableControlRequest.builder()
                .controlIdentifier(controlIdentifier)
                .targetIdentifier(targetIdentifier)
                .build();

        return getAsyncClient().disableControl(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Disable control operation started. Operation
ID: " + operationId);

                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getControlOperationAsync(operationId)
                                    .thenAccept(status -> {
                                        System.out.println("Control operation
status: " + status);

                                        if (status ==
ControlOperationStatus.SUCCEEDED
                                                || status ==
ControlOperationStatus.FAILED) {
                                            resultFuture.complete(operationId);
                                        } else {
                                            // poll again after 30 seconds
                                            CompletableFuture.delayedExecutor(30,
TimeUnit.SECONDS)
                                                    .execute(this);
```

```
                                    }
                                })
                                .exceptionally(ex -> {
                                    resultFuture.completeExceptionally(ex);
                                    return null;
                                });
                    }
                };

                // start polling immediately
                poller.run();

                return resultFuture;
            })
            .exceptionally(ex -> {
                Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                if (cause instanceof ControlTowerException e) {
                    String errorCode = e.awsErrorDetails().errorCode();

                    if ("ResourceNotFoundException".equals(errorCode)) {
                        // SPEC: notify user and continue
                        System.out.println("Control not found for disabling: " +
e.getMessage());
                        return null;
                    }

                    throw new CompletionException(
                            "Error disabling control: " + e.getMessage(), e);
                }

                if (cause instanceof SdkException) {
                    throw new CompletionException(
                            "SDK error disabling control: " +
cause.getMessage(), cause);
                }

                throw new CompletionException(
                        "Failed to disable control", cause);
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DisableControl을 참조하세요.

`EnableBaseline`

다음 코드 예시는 `EnableBaseline`의 사용 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Asynchronously enables a baseline for the specified target if not already
enabled.
    *
    * @param targetIdentifier       The ARN of the target (OU or account).
    * @param baselineIdentifier     The baseline definition ARN to enable.
    * @param baselineVersion        The baseline version to enable.
    * @return A CompletableFuture containing the enabled baseline ARN, or null if
already enabled.
    */
   public CompletableFuture<String> enableBaselineAsync(
           String targetIdentifier,
           String baselineIdentifier,
           String baselineVersion
   ) {
       EnableBaselineRequest request = EnableBaselineRequest.builder()
               .baselineIdentifier(baselineIdentifier)
               .baselineVersion(baselineVersion)
               .targetIdentifier(targetIdentifier)
               .build();

       return getAsyncClient().enableBaseline(request)
               .handle((resp, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                       if (cause instanceof ControlTowerException e) {
```

```
                            String code = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                            String msg = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();

                            if ("ValidationException".equals(code) &&
msg.contains("already enabled")) {
                                System.out.println("Baseline is already enabled for
this target # fetching ARN...");
                                return fetchEnabledBaselineArn(targetIdentifier,
baselineIdentifier)
                                        .join(); // fetch existing ARN synchronously
                            }

                            throw new RuntimeException("Error enabling baseline: " +
code + " - " + msg, e);
                        }

                        throw new RuntimeException("Unexpected error enabling
baseline: " + cause.getMessage(), cause);
                    }

                    return resp;
                })
                .thenCompose(result -> {
                    if (result instanceof EnableBaselineResponse resp) {
                        String operationId = resp.operationIdentifier();
                        String enabledBaselineArn = resp.arn();
                        System.out.println("Baseline enable started. ARN: " +
enabledBaselineArn
                                + ", operation ID: " + operationId);

                        // Inline polling
                        return CompletableFuture.supplyAsync(() -> {
                            while (true) {
                                GetBaselineOperationRequest opReq =
GetBaselineOperationRequest.builder()
                                        .operationIdentifier(operationId)
                                        .build();

                                GetBaselineOperationResponse opResp =
getAsyncClient().getBaselineOperation(opReq).join();
                                BaselineOperation op = opResp.baselineOperation();
                                BaselineOperationStatus status = op.status();
```

```
                                System.out.println("Operation " + operationId + "
status: " + status);

                                if (status == BaselineOperationStatus.SUCCEEDED) {
                                    return enabledBaselineArn;
                                } else if (status == BaselineOperationStatus.FAILED)
{
                                    String opId = op.operationIdentifier();
                                    String reason = op.statusMessage() != null ?
op.statusMessage() : "No failure reason provided";
                                    throw new RuntimeException("Baseline operation
failed (ID: " + opId + "), status: "
                                            + status + ", reason: " + reason);
                                }

                                try {
                                    Thread.sleep(Duration.ofSeconds(15).toMillis());
                                } catch (InterruptedException e) {
                                    Thread.currentThread().interrupt();
                                    throw new RuntimeException(e);
                                }
                            }
                        });
                } else if (result instanceof String existingArn) {
                    // Already enabled branch
                    return CompletableFuture.completedFuture(existingArn);
                }

                return CompletableFuture.completedFuture(null);
            });
    }


    /**
     * Fetches the ARN of an already-enabled baseline for the target asynchronously.
     */
    private CompletableFuture<String> fetchEnabledBaselineArn(String
targetIdentifier, String baselineIdentifier) {
        return
getAsyncClient().listEnabledBaselines(ListEnabledBaselinesRequest.builder().build())
                .thenApply(listResp -> {
                    for (EnabledBaselineSummary eb : listResp.enabledBaselines()) {
                        if (baselineIdentifier.equals(eb.baselineIdentifier())
                                && targetIdentifier.equals(eb.targetIdentifier())) {
```

```
                        return eb.arn();
                    }
                }
                return null; // not yet available
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 EnableBaseline을 참조하세요.

`EnableControl`

다음 코드 예시는 `EnableControl`의 사용 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Enables a control for a specified target.
 *
 * @param controlIdentifier the identifier of the control to enable
 * @param targetIdentifier  the identifier of the target (e.g., OU ARN)
 * @return the operation identifier
 * @throws ControlTowerException if a service-specific error occurs
 * @throws SdkException          if an SDK error occurs
 */
public CompletableFuture<String> enableControlAsync(
        String controlIdentifier,
        String targetIdentifier) {

    EnableControlRequest request = EnableControlRequest.builder()
            .controlIdentifier(controlIdentifier)
            .targetIdentifier(targetIdentifier)
            .build();

    return getAsyncClient().enableControl(request)
```

```
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Enable control operation started. Operation
ID: " + operationId);

                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getControlOperationAsync(operationId)
                                    .thenAccept(status -> {
                                        System.out.println("Control operation
status: " + status);

                                        if (status ==
ControlOperationStatus.SUCCEEDED
                                                || status ==
ControlOperationStatus.FAILED) {
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Poll again after 30 seconds
                                            CompletableFuture.delayedExecutor(30,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start polling immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
```

```
                        String errorCode = e.awsErrorDetails().errorCode();
                        String message = e.getMessage() != null ? e.getMessage() :
"";

                        if ("ValidationException".equals(errorCode)
                                && message.contains("already enabled")) {
                            System.out.println("Control is already enabled for this
target");
                            return null;
                        }

                        if ("ResourceNotFoundException".equals(errorCode)
                                && message.contains("not registered with AWS Control
Tower")) {
                            System.out.println(
                                    "Control Tower must be enabled to work with
controls.");
                            return null;
                        }

                        throw new CompletionException(
                                "Couldn't enable control: %s".formatted(message),
                                e
                        );
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error enabling control: %s"
                                        .formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException(
                            "Failed to enable control",
                            cause
                    );
                });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 EnableControl을 참조하세요.

GetBaselineOperation

다음 코드 예시는 GetBaselineOperation의 사용 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Gets the status of a baseline operation.
 *
 * @param operationIdentifier the identifier of the operation
 * @return the operation status
 * @throws ControlTowerException if a service-specific error occurs
 * @throws SdkException          if an SDK error occurs
 */
public CompletableFuture<BaselineOperationStatus> getBaselineOperationAsync(
        String operationIdentifier) {

    GetBaselineOperationRequest request = GetBaselineOperationRequest.builder()
            .operationIdentifier(operationIdentifier)
            .build();

    return getAsyncClient().getBaselineOperation(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause() != null
                            ? exception.getCause()
                            : exception;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("ResourceNotFoundException".equals(errorCode)) {
                            throw new CompletionException(
                                    "Baseline operation not found: %s"
                                            .formatted(e.getMessage()),
                                    e
```

```
                                );
                            }

                            throw new CompletionException(
                                    "Error getting baseline operation status: %s"
                                            .formatted(e.getMessage()),
                                    e
                            );
                        }

                        if (cause instanceof SdkException) {
                            throw new CompletionException(
                                    "SDK error getting baseline operation status:
%s"
                                            .formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to get baseline operation status",
                                cause
                        );
                    }
                })
                .thenApply(response -> {
                    BaselineOperationStatus status =
                            response.baselineOperation().status();
                    return status;
                });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetBaselineOperation을 참조하세요.

`GetControlOperation`

다음 코드 예시는 `GetControlOperation`의 사용 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Gets the status of a control operation.
     *
     * @param operationIdentifier the identifier of the operation
     * @return the operation status
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<ControlOperationStatus> getControlOperationAsync(
            String operationIdentifier) {

        GetControlOperationRequest request = GetControlOperationRequest.builder()
                .operationIdentifier(operationIdentifier)
                .build();

        return getAsyncClient().getControlOperation(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ControlTowerException e) {
                            String errorCode = e.awsErrorDetails().errorCode();

                            if ("ResourceNotFoundException".equals(errorCode)) {
                                throw new CompletionException(
                                        "Control operation not found:
%s".formatted(e.getMessage()),
                                        e
                                );
                            }

                            throw new CompletionException(
```

```
                                    "Error getting control operation status:
%s".formatted(e.getMessage()),
                                    e
                            );
                        }

                        if (cause instanceof SdkException) {
                            throw new CompletionException(
                                    "SDK error getting control operation status:
%s".formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException("Failed to get control
operation status", cause);
                    }
                })
                .thenApply(response -> response.controlOperation().status());
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetControlOperation을 참조하세요.

`ListBaselines`

다음 코드 예시는 `ListBaselines`의 사용 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Lists all available baselines using pagination to retrieve complete results.
     *
     * @return a list of all baselines
     * @throws ControlTowerException if a service-specific error occurs
```

```
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<BaselineSummary>> listBaselinesAsync() {
        System.out.println("Starting list baselines paginator…");
        ListBaselinesRequest request = ListBaselinesRequest.builder().build();
        ListBaselinesPublisher paginator =
                getAsyncClient().listBaselinesPaginator(request);

        List<BaselineSummary> baselines = new ArrayList<>();
        return paginator.subscribe(response -> {
                    if (response.baselines() != null && !
response.baselines().isEmpty()) {
                        response.baselines().forEach(baseline -> {
                            baselines.add(baseline);
                        });
                    } else {
                        System.out.println("Page contained no baselines.");
                    }
                })
                .thenRun(() ->
                        System.out.println("Successfully listed baselines. Total: "
 + baselines.size())
                )
                .thenApply(v -> baselines)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("AccessDeniedException".equals(errorCode)) {
                            throw new CompletionException(
                                    "Access denied when listing baselines:
 %s".formatted(e.getMessage()),
                                    e
                            );
                        }

                        throw new CompletionException(
                                "Error listing baselines:
 %s".formatted(e.getMessage()),
                                e
                        );
                    }
```

```
                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing baselines:
%s".formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException("Failed to list baselines",
cause);
                });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListBaselines을 참조하세요.

`ListEnabledBaselines`

다음 코드 예시는 `ListEnabledBaselines`의 사용 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Lists all enabled baselines using pagination to retrieve complete results.
     *
     * @return a list of all enabled baselines
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<EnabledBaselineSummary>>
listEnabledBaselinesAsync() {
        System.out.println("Starting list enabled baselines paginator…");

        ListEnabledBaselinesRequest request =
                ListEnabledBaselinesRequest.builder().build();
```

```
        ListEnabledBaselinesPublisher paginator =
                getAsyncClient().listEnabledBaselinesPaginator(request);

        List<EnabledBaselineSummary> enabledBaselines = new ArrayList<>();
        return paginator.subscribe(response -> {
                    if (response.enabledBaselines() != null
                            && !response.enabledBaselines().isEmpty()) {

                        response.enabledBaselines().forEach(baseline -> {
                            enabledBaselines.add(baseline);
                        });
                    } else {
                        System.out.println("Page contained no enabled baselines.");
                    }
                })
                .thenRun(() ->
                        System.out.println(
                                "Successfully listed enabled baselines. Total: " +
                                        enabledBaselines.size()
                        )
                )
                .thenApply(v -> enabledBaselines)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("AccessDeniedException".equals(errorCode)) {
                            throw new CompletionException(
                                    "Access denied when listing enabled baselines:
%s".formatted(e.getMessage()), e);
                        }

                        throw new CompletionException(
                                "Error listing enabled baselines: %s"
                                        .formatted(e.getMessage()),
                                e
                        );
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
```

```
                                "SDK error listing enabled baselines: %s"
                                        .formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException(
                            "Failed to list enabled baselines",
                            cause
                    );
                });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListEnabledBaselines을 참조하세요.

`ListEnabledControls`

다음 코드 예시는 `ListEnabledControls`의 사용 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Lists all enabled controls for a specific target using pagination.
     *
     * @param targetIdentifier the identifier of the target (e.g., OU ARN)
     * @return a list of enabled controls
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<EnabledControlSummary>>
listEnabledControlsAsync(String targetIdentifier) {
        System.out.println("Starting list enabled controls paginator for target " +
targetIdentifier);
        ListEnabledControlsRequest request = ListEnabledControlsRequest.builder()
```

```
                .targetIdentifier(targetIdentifier)
                .build();

        ListEnabledControlsPublisher paginator =
 getAsyncClient().listEnabledControlsPaginator(request);
        List<EnabledControlSummary> enabledControls = new ArrayList<>();

        // Subscribe to the paginator asynchronously
        return paginator.subscribe(response -> {
                    if (response.enabledControls() != null && !
response.enabledControls().isEmpty()) {
                        response.enabledControls().forEach(control -> {
                            enabledControls.add(control);
                        });
                    } else {
                        System.out.println("Page contained no enabled controls.");
                    }
                })
                .thenRun(() -> System.out.println(
                        "Successfully retrieved "+enabledControls.size() +" enabled
 controls for target "+targetIdentifier
                ))
                .thenApply(v -> enabledControls)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        switch (errorCode) {
                            case "AccessDeniedException":
                                throw new CompletionException(
                                        "Access denied when listing enabled
 controls: %s".formatted(e.getMessage()), e);

                            case "ResourceNotFoundException":
                                if (e.getMessage() != null &&
 e.getMessage().contains("not registered with AWS Control Tower")) {
                                    throw new CompletionException(
                                            "Control Tower must be enabled to work
 with controls", e);
                                }
                                throw new CompletionException(
```

```
                                        "Target not found when listing enabled
controls: %s".formatted(e.getMessage()), e);

                            default:
                                throw new CompletionException(
                                        "Error listing enabled controls:
%s".formatted(e.getMessage()), e);
                            }
                        }

                        if (cause instanceof SdkException) {
                            throw new CompletionException(
                                    "SDK error listing enabled controls:
%s".formatted(cause.getMessage()), cause);
                        }

                        throw new CompletionException("Failed to list enabled controls",
cause);
                    });
        }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListEnabledControls을 참조하세요.

`ListLandingZones`

다음 코드 예시는 `ListLandingZones`의 사용 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Lists all landing zones using pagination to retrieve complete results.
     *
     * @return a list of all landing zones
     * @throws ControlTowerException if a service-specific error occurs
```

```
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<LandingZoneSummary>> listLandingZonesAsync() {
        System.out.println("Starting list landing zones paginator…");

        ListLandingZonesRequest request = ListLandingZonesRequest.builder().build();
        ListLandingZonesPublisher paginator =
 getAsyncClient().listLandingZonesPaginator(request);
        List<LandingZoneSummary> landingZones = new ArrayList<>();

        return paginator.subscribe(response -> {
                    if (response.landingZones() != null && !
response.landingZones().isEmpty()) {
                        response.landingZones().forEach(lz -> {
                            System.out.println("Landing zone ARN: " + lz.arn());
                            landingZones.add(lz);
                        });
                    } else {
                        System.out.println("Page contained no landing zones.");
                    }
                })
                .thenRun(() -> System.out.println("Successfully retrieved "+
 landingZones.size() + " landing zones." ))
                .thenApply(v -> landingZones)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();
                        switch (errorCode) {
                            case "AccessDeniedException":
                                throw new CompletionException(
                                        "Access denied when listing landing zones: "
 + e.getMessage(), e);
                            default:
                                throw new CompletionException(
                                        "Error listing landing zones: " +
 e.getMessage(), e);
                        }
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
```

```
                                "SDK error listing landing zones: " +
cause.getMessage(), cause);
                    }

                    throw new CompletionException("Failed to list landing zones",
cause);
                });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListLandingZones을 참조하세요.

`ResetEnabledBaseline`

다음 코드 예시는 `ResetEnabledBaseline`의 사용 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Resets an enabled baseline for a specific target.
     *
     * @param enabledBaselineIdentifier the identifier of the enabled baseline to
reset
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> resetEnabledBaselineAsync(String
enabledBaselineIdentifier) {

        System.out.println("Starting reset of enabled baseline…");
        System.out.println("This operation will check the status every 15 seconds
until it completes (SUCCEEDED or FAILED).");

        ResetEnabledBaselineRequest request = ResetEnabledBaselineRequest.builder()
                .enabledBaselineIdentifier(enabledBaselineIdentifier)
```

```
                .build();

        return getAsyncClient().resetEnabledBaseline(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Reset enabled baseline operation ID: " +
operationId);

                    // Polling loop
                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getBaselineOperationAsync(operationId)
                                    .thenAccept(statusObj -> {
                                        String status = statusObj.toString(); //
Convert enum/status to string for printing
                                        System.out.println("Current baseline
operation status: " + status + " # waiting for SUCCEEDED or FAILED...");

                                        if ("SUCCEEDED".equalsIgnoreCase(status) ||
"FAILED".equalsIgnoreCase(status)) {
                                            System.out.println("Baseline operation
finished with status: " + status);
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Schedule next poll in 15 seconds
                                            CompletableFuture.delayedExecutor(15,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        System.out.println("Error checking baseline
operation status: " + ex.getMessage());
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start first poll immediately
```

```
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                        String errorMessage = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();

                        System.out.println("ControlTowerException caught: Code=" +
errorCode + ", Message=" + errorMessage);
                        return null;
                    }

                    if (cause instanceof SdkException sdkEx) {
                        System.out.println("SDK exception caught: " +
sdkEx.getMessage());
                        return null;
                    }

                    System.out.println("Unexpected exception resetting baseline: " +
cause.getMessage());
                    return null;
                });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ResetEnabledBaseline을 참조하세요.

SDK for Java 2.x를 사용한 Firehose 예제

다음 코드 예제에서는 Firehose와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`PutRecord`

다음 코드 예시는 `PutRecord`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Puts a record to the specified Amazon Kinesis Data Firehose delivery stream.
    *
    * @param record The record to be put to the delivery stream. The record must be
a {@link Map} of String keys and Object values.
    * @param deliveryStreamName The name of the Amazon Kinesis Data Firehose
delivery stream to which the record should be put.
    * @throws IllegalArgumentException if the input record or delivery stream name
is null or empty.
    * @throws RuntimeException if there is an error putting the record to the
delivery stream.
    */
   public static void putRecord(Map<String, Object> record, String
deliveryStreamName) {
       if (record == null || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
```

```
            throw new IllegalArgumentException("Invalid input: record or delivery
stream name cannot be null/empty");
        }
        try {
            String jsonRecord = new ObjectMapper().writeValueAsString(record);
            Record firehoseRecord = Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                .build();

            PutRecordRequest putRecordRequest = PutRecordRequest.builder()
                .deliveryStreamName(deliveryStreamName)
                .record(firehoseRecord)
                .build();

            getFirehoseClient().putRecord(putRecordRequest);
            System.out.println("Record sent: " + jsonRecord);
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record: " + e.getMessage(),
e);
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 PutRecord를 참조하세요.

`PutRecordBatch`

다음 코드 예시는 `PutRecordBatch`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Puts a batch of records to an Amazon Kinesis Data Firehose delivery stream.
     *
     * @param records           a list of maps representing the records to be sent
```

```
    * @param batchSize         the maximum number of records to include in each
batch
    * @param deliveryStreamName the name of the Kinesis Data Firehose delivery
stream
    * @throws IllegalArgumentException if the input parameters are invalid (null or
empty)
    * @throws RuntimeException         if there is an error putting the record
batch
    */
   public static void putRecordBatch(List<Map<String, Object>> records, int
batchSize, String deliveryStreamName) {
       if (records == null || records.isEmpty() || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
           throw new IllegalArgumentException("Invalid input: records or delivery
stream name cannot be null/empty");
       }
       ObjectMapper objectMapper = new ObjectMapper();

       try {
           for (int i = 0; i < records.size(); i += batchSize) {
               List<Map<String, Object>> batch = records.subList(i, Math.min(i +
batchSize, records.size()));

               List<Record> batchRecords = batch.stream().map(record -> {
                   try {
                       String jsonRecord = objectMapper.writeValueAsString(record);
                       return Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                               .build();
                   } catch (Exception e) {
                       throw new RuntimeException("Error creating Firehose record",
e);
                   }
               }).collect(Collectors.toList());

               PutRecordBatchRequest request = PutRecordBatchRequest.builder()
                   .deliveryStreamName(deliveryStreamName)
                   .records(batchRecords)
                   .build();

               PutRecordBatchResponse response =
getFirehoseClient().putRecordBatch(request);
```

```
                if (response.failedPutCount() > 0) {
                    response.requestResponses().stream()
                        .filter(r -> r.errorCode() != null)
                        .forEach(r -> System.err.println("Failed record: " +
r.errorMessage()));
                }
                System.out.println("Batch sent with size: " + batchRecords.size());
            }
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record batch: " +
e.getMessage(), e);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutRecordBatch를 참조하세요.

시나리오

Firehose에 레코드 넣기

다음 코드 예제는 Firehose를 사용하여 개별 레코드 및 배치 레코드를 처리하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예제는 개별 레코드 및 배치 레코드를 Firehose에 보냅니다.

```
/**
 * Amazon Firehose Scenario example using Java V2 SDK.
 *
 * Demonstrates individual and batch record processing,
 * and monitoring Firehose delivery stream metrics.
 */
public class FirehoseScenario {

    private static FirehoseClient firehoseClient;
```

```
    private static CloudWatchClient cloudWatchClient;

    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <deliveryStreamName>
                Where:
                    deliveryStreamName - The Firehose delivery stream name.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String deliveryStreamName = args[0];

        try {
            // Read and parse sample data.
            String jsonContent = readJsonFile("sample_records.json");
            ObjectMapper objectMapper = new ObjectMapper();
            List<Map<String, Object>> sampleData =
objectMapper.readValue(jsonContent, new TypeReference<>() {});

            // Process individual records.
            System.out.println("Processing individual records...");
            sampleData.subList(0, 100).forEach(record -> {
                try {
                    putRecord(record, deliveryStreamName);
                } catch (Exception e) {
                    System.err.println("Error processing record: " +
e.getMessage());
                }
            });

            // Monitor metrics.
            monitorMetrics(deliveryStreamName);

            // Process batch records.
            System.out.println("Processing batch records...");
            putRecordBatch(sampleData.subList(100, sampleData.size()), 500,
deliveryStreamName);
            monitorMetrics(deliveryStreamName);
```

```
        } catch (Exception e) {
            System.err.println("Scenario failed: " + e.getMessage());
        } finally {
            closeClients();
        }
    }

    private static FirehoseClient getFirehoseClient() {
        if (firehoseClient == null) {
            firehoseClient = FirehoseClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return firehoseClient;
    }

    private static CloudWatchClient getCloudWatchClient() {
        if (cloudWatchClient == null) {
            cloudWatchClient = CloudWatchClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return cloudWatchClient;
    }

    /**
     * Puts a record to the specified Amazon Kinesis Data Firehose delivery stream.
     *
     * @param record The record to be put to the delivery stream. The record must be
a {@link Map} of String keys and Object values.
     * @param deliveryStreamName The name of the Amazon Kinesis Data Firehose
delivery stream to which the record should be put.
     * @throws IllegalArgumentException if the input record or delivery stream name
is null or empty.
     * @throws RuntimeException if there is an error putting the record to the
delivery stream.
     */
    public static void putRecord(Map<String, Object> record, String
deliveryStreamName) {
        if (record == null || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
            throw new IllegalArgumentException("Invalid input: record or delivery
stream name cannot be null/empty");
        }
```

```
        try {
            String jsonRecord = new ObjectMapper().writeValueAsString(record);
            Record firehoseRecord = Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                .build();

            PutRecordRequest putRecordRequest = PutRecordRequest.builder()
                .deliveryStreamName(deliveryStreamName)
                .record(firehoseRecord)
                .build();

            getFirehoseClient().putRecord(putRecordRequest);
            System.out.println("Record sent: " + jsonRecord);
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record: " + e.getMessage(),
e);
        }
    }


    /**
     * Puts a batch of records to an Amazon Kinesis Data Firehose delivery stream.
     *
     * @param records           a list of maps representing the records to be sent
     * @param batchSize         the maximum number of records to include in each
batch
     * @param deliveryStreamName the name of the Kinesis Data Firehose delivery
stream
     * @throws IllegalArgumentException if the input parameters are invalid (null or
empty)
     * @throws RuntimeException         if there is an error putting the record
batch
     */
    public static void putRecordBatch(List<Map<String, Object>> records, int
batchSize, String deliveryStreamName) {
        if (records == null || records.isEmpty() || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
            throw new IllegalArgumentException("Invalid input: records or delivery
stream name cannot be null/empty");
        }
        ObjectMapper objectMapper = new ObjectMapper();

        try {
```

```
            for (int i = 0; i < records.size(); i += batchSize) {
                List<Map<String, Object>> batch = records.subList(i, Math.min(i +
batchSize, records.size()));

                List<Record> batchRecords = batch.stream().map(record -> {
                    try {
                        String jsonRecord = objectMapper.writeValueAsString(record);
                        return Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                                .build();
                    } catch (Exception e) {
                        throw new RuntimeException("Error creating Firehose record",
e);
                    }
                }).collect(Collectors.toList());

                PutRecordBatchRequest request = PutRecordBatchRequest.builder()
                    .deliveryStreamName(deliveryStreamName)
                    .records(batchRecords)
                    .build();

                PutRecordBatchResponse response =
getFirehoseClient().putRecordBatch(request);

                if (response.failedPutCount() > 0) {
                    response.requestResponses().stream()
                        .filter(r -> r.errorCode() != null)
                        .forEach(r -> System.err.println("Failed record: " +
r.errorMessage()));
                }
                System.out.println("Batch sent with size: " + batchRecords.size());
            }
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record batch: " +
e.getMessage(), e);
        }
    }

    public static void monitorMetrics(String deliveryStreamName) {
        Instant endTime = Instant.now();
        Instant startTime = endTime.minusSeconds(600);
```

```
        List<String> metrics = List.of("IncomingBytes", "IncomingRecords",
"FailedPutCount");
        metrics.forEach(metric -> monitorMetric(metric, startTime, endTime,
deliveryStreamName));
    }

    private static void monitorMetric(String metricName, Instant startTime, Instant
endTime, String deliveryStreamName) {
        try {
            GetMetricStatisticsRequest request =
GetMetricStatisticsRequest.builder()
                .namespace("AWS/Firehose")
                .metricName(metricName)

.dimensions(Dimension.builder().name("DeliveryStreamName").value(deliveryStreamName).build(
                .startTime(startTime)
                .endTime(endTime)
                .period(60)
                .statistics(Statistic.SUM)
                .build();

            GetMetricStatisticsResponse response =
getCloudWatchClient().getMetricStatistics(request);
            double totalSum =
response.datapoints().stream().mapToDouble(Datapoint::sum).sum();
            System.out.println(metricName + ": " + totalSum);
        } catch (Exception e) {
            System.err.println("Failed to monitor metric " + metricName + ": " +
e.getMessage());
        }
    }

    public static String readJsonFile(String fileName) throws IOException {
        try (InputStream inputStream =
FirehoseScenario.class.getResourceAsStream("/" + fileName);
             Scanner scanner = new Scanner(inputStream, StandardCharsets.UTF_8)) {
            return scanner.useDelimiter("\\\\A").next();
        } catch (Exception e) {
            throw new RuntimeException("Error reading file: " + fileName, e);
        }
    }

    private static void closeClients() {
        try {
```

```
            if (firehoseClient != null) firehoseClient.close();
            if (cloudWatchClient != null) cloudWatchClient.close();
        } catch (Exception e) {
            System.err.println("Error closing clients: " + e.getMessage());
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - PutRecord
  - PutRecordBatch

SDK for Java 2.x를 사용한 Amazon DocumentDB 예제

다음 코드 예제에서는 AWS SDK for Java 2.x Amazon DocumentDB에서를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 서버리스 예제

서버리스 예제

Amazon DocumentDB 트리거에서 간접적으로 Lambda 함수 호출

다음 코드 예제에서는 DocumentDB 변경 스트림에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 DocumentDB 페이로드를 검색하고 레코드 콘텐츠를 로깅합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Java를 사용하여 Lambda로 Amazon DocumentDB 이벤트 소비

```
import java.util.List;
import java.util.Map;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;

public class Example implements RequestHandler<Map<String, Object>, String> {

    @SuppressWarnings("unchecked")
    @Override
    public String handleRequest(Map<String, Object> event, Context context) {
        List<Map<String, Object>> events = (List<Map<String, Object>>)
 event.get("events");
        for (Map<String, Object> record : events) {
            Map<String, Object> eventData = (Map<String, Object>)
 record.get("event");
            processEventData(eventData);
        }

        return "OK";
    }

    @SuppressWarnings("unchecked")
    private void processEventData(Map<String, Object> eventData) {
        String operationType = (String) eventData.get("operationType");
        System.out.println("operationType: %s".formatted(operationType));

        Map<String, Object> ns = (Map<String, Object>) eventData.get("ns");

        String db = (String) ns.get("db");
        System.out.println("db: %s".formatted(db));
        String coll = (String) ns.get("coll");
        System.out.println("coll: %s".formatted(coll));

        Map<String, Object> fullDocument = (Map<String, Object>)
 eventData.get("fullDocument");
        System.out.println("fullDocument: %s".formatted(fullDocument));
    }

}
```

Java 2.x용 SDK를 사용하는 DynamoDB 예제

다음 코드 예제에서는 DynamoDB와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

AWS 커뮤니티 기여는 여러 팀이 생성하고 유지 관리하는 예입니다 AWS. 피드백을 제공하려면 연결된 리포지토리에 제공된 메커니즘을 사용합니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시작하기

Hello DynamoDB

다음 코드 예제는 DynamoDB를 사용하여 시작하는 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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!");
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListTables를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 영화 데이터를 저장할 수 있는 테이블을 생성합니다.

- 테이블에 하나의 영화를 추가하고 가져오고 업데이트합니다.
- 샘플 JSON 파일에서 테이블에 영화 데이터를 씁니다.
- 특정 연도에 개봉된 영화를 쿼리합니다.
- 특정 연도 범위 동안 개봉된 영화를 스캔합니다.
- 테이블에서 영화를 삭제한 다음, 테이블을 삭제합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

DynamoDB 테이블을 생성합니다.

```
// 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)
            .billingMode(BillingMode.PAY_PER_REQUEST) //  DynamoDB automatically
scales based on traffic.
            .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);
        }
    }
```

헬퍼 함수를 생성하여 샘플 JSON 파일을 다운로드하고 추출합니다.

```
    // 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++;
        }
    }
```

테이블에서 항목을 가져옵니다.

```
    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);
        }
    }
```

전체 예제는 다음과 같습니다.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 * <p>
 * This Java example performs these tasks:
 * <p>
 * 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 {
        String tableName = "Movies";
        String fileName = "../../../resources/sample_files/movies.json";
        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)
            .billingMode(BillingMode.PAY_PER_REQUEST) //  DynamoDB automatically
scales based on traffic.
            .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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

작업

`BatchGetItem`

다음 코드 예시는 `BatchGetItem`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

서비스 클라이언트를 사용하여 배치 항목을 가져오는 방법을 보여줍니다.

```
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.");
        }
    }
}
```

서비스 클라이언트와 페이지네이터를 사용하여 배치 항목을 가져오는 방법을 보여줍니다.

```
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());
            });
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 BatchGetItem을 참조하세요.

`BatchWriteItem`

다음 코드 예시는 `BatchWriteItem`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

서비스 클라이언트를 사용하여 테이블에 많은 항목을 삽입합니다.

```
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);
        }
    }
}
```

향상된 클라이언트를 사용하여 테이블에 많은 항목을 삽입합니다.

```
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);
                 }
         }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 BatchWriteItem을 참조하세요.

`CreateTable`

다음 코드 예시는 `CreateTable`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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.BillingMode;
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.OnDemandThroughput;
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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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())
            .billingMode(BillingMode.PAY_PER_REQUEST) //  DynamoDB automatically
scales based on traffic.
            .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 "";
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateTable을 참조하세요.

`DeleteItem`

다음 코드 예시는 `DeleteItem`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteItem을 참조하세요.

`DeleteTable`

다음 코드 예시는 `DeleteTable`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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!");
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteTable을 참조하세요.

`DescribeTable`

다음 코드 예시는 `DescribeTable`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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!");
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeTable을 참조하세요.

`DescribeTimeToLive`

다음 코드 예시는 `DescribeTimeToLive`의 사용 방법을 보여줍니다.

SDK for Java 2.x

AWS SDK for Java 2.x를 사용하여 기존 DynamoDB 테이블의 TTL 구성을 설명합니다.

```
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.logging.Level;
import java.util.logging.Logger;

    public DescribeTimeToLiveResponse describeTTL(final String tableName, final
 Region region) {
        final DescribeTimeToLiveRequest request =
            DescribeTimeToLiveRequest.builder().tableName(tableName).build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            return ddb.describeTimeToLive(request);
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeTimeToLive를 참조하세요.

`GetItem`

다음 코드 예시는 `GetItem`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

DynamoDbClient를 사용하여 테이블에서 항목을 가져옵니다.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetItem을 참조하세요.

`ListTables`

다음 코드 예시는 `ListTables`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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!");
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListTables를 참조하세요.

PutItem

다음 코드 예시는 PutItem의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

DynamoDbClient를 사용하여 테이블에 항목 추가

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutItem을 참조하세요.

Query

다음 코드 예시는 Query의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

DynamoDbClient를 사용하여 테이블을 쿼리합니다.

```
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;
    }
}
```

DynamoDbClient 및 보조 인덱스를 사용하여 테이블을 쿼리합니다.

```
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);
        }
```

```
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Query를 참조하세요.

Scan

다음 코드 예시는 Scan의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

DynamoDbClient를 사용하여 Amazon DynamoDB 테이블을 스캔합니다.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Scan을 참조하세요.

`UpdateItem`

다음 코드 예시는 `UpdateItem`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

DynamoDbClient를 사용하여 테이블의 항목 업데이트

```
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!");
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateItem을 참조하세요.

`UpdateTimeToLive`

다음 코드 예시는 `UpdateTimeToLive`의 사용 방법을 보여줍니다.

SDK for Java 2.x

AWS SDK for Java 2.x를 사용하여 기존 DynamoDB 테이블에서 TTL을 활성화합니다.

```
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.logging.Level;
import java.util.logging.Logger;

    public UpdateTimeToLiveResponse enableTTL(final String tableName, final String
 attributeName, final Region region) {
        final TimeToLiveSpecification ttlSpec = TimeToLiveSpecification.builder()
            .attributeName(attributeName)
            .enabled(true)
            .build();

        final UpdateTimeToLiveRequest request = UpdateTimeToLiveRequest.builder()
            .tableName(tableName)
            .timeToLiveSpecification(ttlSpec)
            .build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            return ddb.updateTimeToLive(request);
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
```

AWS SDK for Java 2.x를 사용하여 기존 DynamoDB 테이블에서 TTL을 비활성화합니다.

```
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.logging.Level;
import java.util.logging.Logger;

    public UpdateTimeToLiveResponse disableTTL(
        final String tableName, final String attributeName, final Region region) {
        final TimeToLiveSpecification ttlSpec = TimeToLiveSpecification.builder()
            .attributeName(attributeName)
            .enabled(false)
            .build();

        final UpdateTimeToLiveRequest request = UpdateTimeToLiveRequest.builder()
            .tableName(tableName)
            .timeToLiveSpecification(ttlSpec)
            .build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            return ddb.updateTimeToLive(request);
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateTimeToLive를 참조하세요.

시나리오

DynamoDB 테이블에 데이터를 제출하기 위한 앱 구축

다음 코드 예제에서는 Amazon DynamoDB 테이블에 데이터를 제출하고 사용자가 테이블을 업데이트하면 알려주는 애플리케이션을 빌드하는 방법을 보여줍니다.

SDK for Java 2.x

Amazon DynamoDB Java API를 사용하여 데이터를 제출하고 Amazon Simple Notification Service Java API를 사용하여 문자 메시지를 전송하는 동적 웹 애플리케이션을 생성하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- DynamoDB
- Amazon SNS

단일 속성과 여러 값 비교

다음 코드 예제에서는 DynamoDB에서 여러 값을 단일 속성과 비교하는 방법을 보여줍니다.

- IN 연산자를 사용하여 단일 속성과 여러 값을 비교합니다.
- IN 연산자를 여러 OR 조건과 비교합니다.
- IN을 사용할 때 성능 및 표현식 복잡성 측면의 이점을 이해합니다.

SDK for Java 2.x

를 사용하여 여러 값을 DynamoDB의 단일 속성과 비교합니다 AWS SDK for Java 2.x.

```
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.ScanRequest;
import software.amazon.awssdk.services.dynamodb.model.ScanResponse;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Locale;
import java.util.Map;

    /**
     * Queries a table using the IN operator to compare multiple values with a
 single attribute.
```

```
     *
     * <p>This method demonstrates how to use the IN operator in a filter expression
     * to match an attribute against multiple values.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param partitionKeyName The name of the partition key attribute
     * @param partitionKeyValue The value of the partition key to query
     * @param attributeName The name of the attribute to compare
     * @param valuesList List of values to compare against
     * @return The query response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static QueryResponse compareMultipleValues(
        DynamoDbClient dynamoDbClient,
        String tableName,
        String partitionKeyName,
        AttributeValue partitionKeyValue,
        String attributeName,
        List<AttributeValue> valuesList) {

        // Create expression attribute names
        Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#pkName", partitionKeyName);
        expressionAttributeNames.put("#attrName", attributeName);

        // Create expression attribute values
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
        expressionAttributeValues.put(":pkValue", partitionKeyValue);

        // Add values for IN operator
        for (int i = 0; i < valuesList.size(); i++) {
            expressionAttributeValues.put(":val" + i, valuesList.get(i));
        }

        // Build the IN clause
        StringBuilder inClause = new StringBuilder();
        for (int i = 0; i < valuesList.size(); i++) {
            if (i > 0) {
                inClause.append(", ");
            }
            inClause.append(":val").append(i);
        }
```

```
        // Define the query parameters
        QueryRequest request = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression("#pkName = :pkValue")
            .filterExpression("#attrName IN (" + inClause.toString() + ")")
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the query operation
        return dynamoDbClient.query(request);
    }

    /**
     * Queries a table using multiple OR conditions to compare multiple values with
a single attribute.
     *
     * <p>This method demonstrates the alternative approach to using the IN
operator,
     * by using multiple OR conditions.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param partitionKeyName The name of the partition key attribute
     * @param partitionKeyValue The value of the partition key to query
     * @param attributeName The name of the attribute to compare
     * @param valuesList List of values to compare against
     * @return The query response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static QueryResponse compareWithOrConditions(
        DynamoDbClient dynamoDbClient,
        String tableName,
        String partitionKeyName,
        AttributeValue partitionKeyValue,
        String attributeName,
        List<AttributeValue> valuesList) {

        // Create expression attribute names
        Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#pkName", partitionKeyName);
        expressionAttributeNames.put("#attrName", attributeName);

        // Create expression attribute values
```

```
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
        expressionAttributeValues.put(":pkValue", partitionKeyValue);

        // Add values for OR conditions
        for (int i = 0; i < valuesList.size(); i++) {
            expressionAttributeValues.put(":val" + i, valuesList.get(i));
        }

        // Build the OR conditions
        StringBuilder orConditions = new StringBuilder();
        for (int i = 0; i < valuesList.size(); i++) {
            if (i > 0) {
                orConditions.append(" OR ");
            }
            orConditions.append("#attrName = :val").append(i);
        }

        // Define the query parameters
        QueryRequest request = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression("#pkName = :pkValue")
            .filterExpression(orConditions.toString())
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the query operation
        return dynamoDbClient.query(request);
    }

    /**
     * Compares the performance of using the IN operator versus multiple OR
conditions.
     *
     * <p>This method demonstrates the performance difference between using the IN
operator
     * and using multiple OR conditions.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param partitionKeyName The name of the partition key attribute
     * @param partitionKeyValue The value of the partition key to query
     * @param attributeName The name of the attribute to compare
     * @param valuesList List of values to compare against
```

```
     * @return Map containing the performance comparison results
     */
    public static Map<String, Object> comparePerformance(
        DynamoDbClient dynamoDbClient,
        String tableName,
        String partitionKeyName,
        AttributeValue partitionKeyValue,
        String attributeName,
        List<AttributeValue> valuesList) {

        Map<String, Object> results = new HashMap<>();

        try {
            // Measure performance of IN operator
            long inStartTime = System.nanoTime();
            QueryResponse inResponse = compareMultipleValues(
                dynamoDbClient, tableName, partitionKeyName, partitionKeyValue,
attributeName, valuesList);
            long inEndTime = System.nanoTime();
            long inDuration = inEndTime - inStartTime;

            // Measure performance of OR conditions
            long orStartTime = System.nanoTime();
            QueryResponse orResponse = compareWithOrConditions(
                dynamoDbClient, tableName, partitionKeyName, partitionKeyValue,
attributeName, valuesList);
            long orEndTime = System.nanoTime();
            long orDuration = orEndTime - orStartTime;

            // Record results
            results.put("inOperatorDuration", inDuration);
            results.put("orConditionsDuration", orDuration);
            results.put("inOperatorItems", inResponse.count());
            results.put("orConditionsItems", orResponse.count());
            results.put("inOperatorExpression", "IN operator with " +
valuesList.size() + " values");
            results.put("orConditionsExpression", valuesList.size() + " OR
conditions");
            results.put("success", true);

        } catch (DynamoDbException e) {
            results.put("success", false);
            results.put("error", e.getMessage());
        }
```

```
        return results;
    }

    /**
     * Scans a table using the IN operator with a large number of values.
     *
     * <p>This method demonstrates how to use the IN operator with a large number of
values,
     * which can help stay within the 300 operator limit.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param attributeName The name of the attribute to compare
     * @param valuesList List of values to compare against
     * @return The scan response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static ScanResponse scanWithLargeInClause(
        DynamoDbClient dynamoDbClient, String tableName, String attributeName,
List<AttributeValue> valuesList) {

        // Create expression attribute names
        Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#attrName", attributeName);

        // Create expression attribute values
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();

        // Add values for IN operator
        for (int i = 0; i < valuesList.size(); i++) {
            expressionAttributeValues.put(":val" + i, valuesList.get(i));
        }

        // Build the IN clause
        StringBuilder inClause = new StringBuilder();
        for (int i = 0; i < valuesList.size(); i++) {
            if (i > 0) {
                inClause.append(", ");
            }
            inClause.append(":val").append(i);
        }

        // Define the scan parameters
```

```
        ScanRequest request = ScanRequest.builder()
            .tableName(tableName)
            .filterExpression("#attrName IN (" + inClause.toString() + ")")
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the scan operation
        return dynamoDbClient.scan(request);
    }

    /**
     * Generates a list of sample values for testing.
     *
     * <p>Helper method to generate a list of sample values for testing.
     *
     * @param valueType The type of values to generate (string, number, or boolean)
     * @param count The number of values to generate
     * @return List of generated attribute values
     */
    public static List<AttributeValue> generateSampleValues(String valueType, int
count) {
        List<AttributeValue> values = new ArrayList<>();

        for (int i = 0; i < count; i++) {
            AttributeValue value;

            switch (valueType.toLowerCase(Locale.ROOT)) {
                case "string":
                    value = AttributeValue.builder().s("Value" + i).build();
                    break;
                case "number":
                    value = AttributeValue.builder().n(String.valueOf(i)).build();
                    break;
                case "boolean":
                    value = AttributeValue.builder().bool(i % 2 == 0).build();
                    break;
                default:
                    throw new IllegalArgumentException("Unsupported value type: " +
valueType);
            }

            values.add(value);
        }
```

```
        return values;
    }
```

여러 값을와 비교하는 예제 사용 AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        System.out.println("Demonstrating how to compare multiple values with a
single attribute in DynamoDB");

        try {
            // Example 1: Using the IN operator
            System.out.println("\nExample 1: Using the IN operator");
            List<AttributeValue> categories = List.of(
                AttributeValue.builder().s("Electronics").build(),
                AttributeValue.builder().s("Computers").build(),
                AttributeValue.builder().s("Accessories").build());

            QueryResponse inResponse = compareMultipleValues(
                dynamoDbClient,
                tableName,
                "Department",
                AttributeValue.builder().s("Retail").build(),
                "Category",
                categories);

            System.out.println("Found " + inResponse.count() + " items using IN
operator");
            System.out.println("Items: " + inResponse.items());

            // Example 2: Using multiple OR conditions
            System.out.println("\nExample 2: Using multiple OR conditions");
            QueryResponse orResponse = compareWithOrConditions(
                dynamoDbClient,
                tableName,
                "Department",
                AttributeValue.builder().s("Retail").build(),
                "Category",
                categories);
```

```
            System.out.println("Found " + orResponse.count() + " items using OR
conditions");
            System.out.println("Items: " + orResponse.items());

            // Example 3: Performance comparison
            System.out.println("\nExample 3: Performance comparison");
            Map<String, Object> perfComparison = comparePerformance(
                dynamoDbClient,
                tableName,
                "Department",
                AttributeValue.builder().s("Retail").build(),
                "Category",
                categories);

            if ((boolean) perfComparison.get("success")) {
                System.out.println("IN operator duration: " +
perfComparison.get("inOperatorDuration") + " ns");
                System.out.println("OR conditions duration: " +
perfComparison.get("orConditionsDuration") + " ns");
                System.out.println("IN operator found " +
perfComparison.get("inOperatorItems") + " items");
                System.out.println("OR conditions found " +
perfComparison.get("orConditionsItems") + " items");
                System.out.println("Expression complexity comparison:");
                System.out.println("  IN operator: " +
perfComparison.get("inOperatorExpression"));
                System.out.println("  OR conditions: " +
perfComparison.get("orConditionsExpression"));
            } else {
                System.out.println("Performance comparison failed: " +
perfComparison.get("error"));
            }

            // Example 4: Using IN with a large number of values
            System.out.println("\nExample 4: Using IN with a large number of
values");
            List<AttributeValue> productIds = generateSampleValues("string", 20);

            ScanResponse largeInResponse = scanWithLargeInClause(dynamoDbClient,
tableName, "ProductId", productIds);

            System.out.println(
                "Found " + largeInResponse.count() + " items using IN with " +
productIds.size() + " values");
```

```
            // Explain the benefits of using IN
            System.out.println("\nKey points about using the IN operator in
DynamoDB:");
            System.out.println("1. The IN operator allows comparing a single
attribute against multiple values");
            System.out.println("2. IN is more concise than using multiple OR
conditions");
            System.out.println("3. IN counts as only 1 operator regardless of the
number of values");
            System.out.println("4. Multiple OR conditions count as 1 operator per
condition plus 1 per OR");
            System.out.println("5. Using IN helps stay within the 300 operator limit
for complex expressions");
            System.out.println("6. IN can be used in filter expressions and
condition expressions");
            System.out.println("7. The IN operator supports up to 100 comparison
values");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - Query
  - Scan

항목의 TTL을 조건부로 업데이트

다음 코드 예제에서는 항목의 TTL을 조건부로 업데이트하는 방법을 보여줍니다.

SDK for Java 2.x

조건을 사용하여 테이블의 기존 DynamoDB 항목에서 TTL을 업데이트합니다.

```
package com.amazon.samplelib.ttl;

import com.amazon.samplelib.CodeSampleUtils;
```

```
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.ConditionalCheckFailedException;
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 java.util.Map;
import java.util.Optional;

/**
 * Updates an item in a DynamoDB table with TTL attributes using a conditional
 expression.
 * This class demonstrates how to conditionally update TTL expiration timestamps.
 */
public class UpdateTTLConditional {

    private static 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)
            """;
    private static final int DAYS_TO_EXPIRE = 90;
    private static final int SECONDS_PER_DAY = 24 * 60 * 60;
    private static final String PRIMARY_KEY_ATTR = "primaryKey";
    private static final String SORT_KEY_ATTR = "sortKey";
    private static final String UPDATED_AT_ATTR = "updatedAt";
    private static final String EXPIRE_AT_ATTR = "expireAt";
    private static final String UPDATE_EXPRESSION = "SET " + UPDATED_AT_ATTR + "=:c,
 " + EXPIRE_AT_ATTR + "=:e";
    private static final String CONDITION_EXPRESSION = "attribute_exists(" +
 PRIMARY_KEY_ATTR + ")";
```

```
   private static final String SUCCESS_MESSAGE = "%s UpdateItem operation with TTL
successful.";
   private static final String CONDITION_FAILED_MESSAGE = "Condition check failed.
Item does not exist.";
   private static final String TABLE_NOT_FOUND_ERROR = "Error: The Amazon DynamoDB
table \"%s\" can't be found.";

   private final DynamoDbClient dynamoDbClient;

   /**
    * Constructs an UpdateTTLConditional with a default DynamoDB client.
    */
   public UpdateTTLConditional() {
       this.dynamoDbClient = null;
   }

   /**
    * Constructs an UpdateTTLConditional with the specified DynamoDB client.
    *
    * @param dynamoDbClient The DynamoDB client to use
    */
   public UpdateTTLConditional(final DynamoDbClient dynamoDbClient) {
       this.dynamoDbClient = dynamoDbClient;
   }

   /**
    * Main method to demonstrate conditionally updating an item with TTL.
    *
    * @param args Command line arguments
    */
   public static void main(final String[] args) {
       try {
           int result = new UpdateTTLConditional().processArgs(args);
           System.exit(result);
       } catch (Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }

   /**
    * Process command line arguments and conditionally update an item with TTL.
    *
    * @param args Command line arguments
```

```
     * @return 0 if successful, non-zero otherwise
     * @throws ResourceNotFoundException If the table doesn't exist
     * @throws DynamoDbException If an error occurs during the operation
     * @throws IllegalArgumentException If arguments are invalid
     */
    public int processArgs(final String[] args) {
        // Argument validation (remove or replace this line when reusing this code)
        CodeSampleUtils.validateArgs(args, new int[] {3, 4}, USAGE);

        final String tableName = args[0];
        final String primaryKey = args[1];
        final String sortKey = args[2];
        final Region region = Optional.ofNullable(args.length > 3 ? args[3] : null)
            .map(Region::of)
            .orElse(Region.US_EAST_1);

        // 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 + (DAYS_TO_EXPIRE * SECONDS_PER_DAY);

        // Create the key map for the item to update
        final Map<String, AttributeValue> keyMap = Map.of(
            PRIMARY_KEY_ATTR, AttributeValue.builder().s(primaryKey).build(),
            SORT_KEY_ATTR, AttributeValue.builder().s(sortKey).build());

        // Create the expression attribute values
        final Map<String, AttributeValue> expressionAttributeValues = Map.of(
            ":c", AttributeValue.builder().n(String.valueOf(currentTime)).build(),
            ":e", AttributeValue.builder().n(String.valueOf(expireDate)).build());

        final UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(keyMap)
            .updateExpression(UPDATE_EXPRESSION)
            .conditionExpression(CONDITION_EXPRESSION)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            final UpdateItemResponse response = ddb.updateItem(request);
```

```
            System.out.println(String.format(SUCCESS_MESSAGE, tableName));
            return 0;
        } catch (ConditionalCheckFailedException e) {
            System.err.println(CONDITION_FAILED_MESSAGE);
            throw e;
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateItem을 참조하세요.

표현식 연산자 수 계산

다음 코드 예제에서는 DynamoDB에서 표현식 연산자 수를 계산하는 방법을 보여줍니다.

- DynamoDB의 연산자 제한(300개)을 이해합니다.
- 복잡한 표현식에서 연산자 수를 계산합니다.
- 제한 범위를 벗어나지 않도록 표현식을 최적화합니다.

SDK for Java 2.x

AWS SDK for Java 2.x를 사용한 표현식 연산자 수 계산을 보여줍니다.

```
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.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
```

```
    /**
     * Creates a complex filter expression with a specified number of conditions.
     *
     * <p>This method demonstrates how to generate a complex expression with
     * a specific number of operators to test the 300 operator limit.
     *
     * @param conditionsCount Number of conditions to include
     * @param useAnd Whether to use AND (true) or OR (false) between conditions
     * @return Map containing the filter expression, attribute values, and operator
count
     */
    public static Map<String, Object> createComplexFilterExpression(int
conditionsCount, boolean useAnd) {
        // Initialize the expression parts and attribute values
        StringBuilder filterExpression = new StringBuilder();
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();

        // Generate the specified number of conditions
        for (int i = 0; i < conditionsCount; i++) {
            // Add the operator between conditions (except for the first one)
            if (i > 0) {
                filterExpression.append(useAnd ? " AND " : " OR ");
            }

            // Alternate between different comparison operators for variety
            String valueKey = ":val" + i;

            switch (i % 5) {
                case 0:
                    filterExpression.append("attribute").append(i).append(" =
").append(valueKey);
                    expressionAttributeValues.put(
                        valueKey, AttributeValue.builder().s("value" + i).build());
                    break;
                case 1:
                    filterExpression.append("attribute").append(i).append(" >
").append(valueKey);
                    expressionAttributeValues.put(
                        valueKey,
AttributeValue.builder().n(String.valueOf(i)).build());
                    break;
                case 2:
                    filterExpression.append("attribute").append(i).append(" <
").append(valueKey);
```

```
                    expressionAttributeValues.put(
                        valueKey,
                        AttributeValue.builder().n(String.valueOf(i * 10)).build());
                    break;
                case 3:
                    filterExpression
                        .append("contains(attribute")
                        .append(i)
                        .append(", ")
                        .append(valueKey)
                        .append(")");
                    expressionAttributeValues.put(
                        valueKey, AttributeValue.builder().s("substring" +
i).build());
                    break;
                case 4:
                    filterExpression
                        .append("attribute_exists(attribute")
                        .append(i)
                        .append(")");
                    break;
                default:
                    // This case will never be reached, but added to satisfy
checkstyle
                    break;
            }
        }

        // Calculate the operator count
        // Each condition has 1 operator (=, >, <, contains, attribute_exists)
        // Each AND or OR between conditions is 1 operator
        int operatorCount = conditionsCount + (conditionsCount > 0 ? conditionsCount
- 1 : 0);

        // Create the result map
        Map<String, Object> result = new HashMap<>();
        result.put("filterExpression", filterExpression.toString());
        result.put("expressionAttributeValues", expressionAttributeValues);
        result.put("operatorCount", operatorCount);

        return result;
    }

    /**
```

```
    * Creates a complex update expression with a specified number of operations.
    *
    * <p>This method demonstrates how to generate a complex update expression with
    * a specific number of operators to test the 300 operator limit.
    *
    * @param operationsCount Number of operations to include
    * @return Map containing the update expression, attribute values, and operator
count
    */
   public static Map<String, Object> createComplexUpdateExpression(int
operationsCount) {
       // Initialize the expression parts and attribute values
       StringBuilder updateExpression = new StringBuilder("SET ");
       Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();

       // Generate the specified number of SET operations
       for (int i = 0; i < operationsCount; i++) {
           // Add comma between operations (except for the first one)
           if (i > 0) {
               updateExpression.append(", ");
           }

           // Alternate between different types of SET operations
           String valueKey = ":val" + i;

           switch (i % 3) {
               case 0:
                   // Simple assignment (1 operator: =)
                   updateExpression.append("attribute").append(i).append(" =
").append(valueKey);
                   expressionAttributeValues.put(
                       valueKey, AttributeValue.builder().s("value" + i).build());
                   break;
               case 1:
                   // Addition (2 operators: = and +)
                   updateExpression
                       .append("attribute")
                       .append(i)
                       .append(" = attribute")
                       .append(i)
                       .append(" + ")
                       .append(valueKey);
                   expressionAttributeValues.put(
```

```
                        valueKey,
AttributeValue.builder().n(String.valueOf(i)).build());
                    break;
                case 2:
                    // Conditional assignment with if_not_exists (2 operators: = and
if_not_exists)
                    updateExpression
                        .append("attribute")
                        .append(i)
                        .append(" = if_not_exists(attribute")
                        .append(i)
                        .append(", ")
                        .append(valueKey)
                        .append(")");
                    expressionAttributeValues.put(
                        valueKey,
                        AttributeValue.builder().n(String.valueOf(i * 10)).build());
                    break;
                default:
                    // This case will never be reached, but added to satisfy
checkstyle
                    break;
            }
        }

        // Calculate the operator count
        // Each operation has 1-2 operators as noted above
        int operatorCount = 0;
        for (int i = 0; i < operationsCount; i++) {
            operatorCount += (i % 3 == 0) ? 1 : 2;
        }

        // Create the result map
        Map<String, Object> result = new HashMap<>();
        result.put("updateExpression", updateExpression.toString());
        result.put("expressionAttributeValues", expressionAttributeValues);
        result.put("operatorCount", operatorCount);

        return result;
    }

    /**
     * Test the operator limit by attempting an operation with a complex expression.
     *
```

```
     * <p>This method demonstrates what happens when an expression approaches or
     * exceeds the 300 operator limit.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param operatorCount Target number of operators to include
     * @return Map containing the result of the operation attempt
     */
    public static Map<String, Object> testOperatorLimit(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, int operatorCount) {

        // Create a complex update expression with the specified operator count
        Map<String, Object> expressionData =
            createComplexUpdateExpression((int) Math.ceil(operatorCount / 1.5)); //
Adjust to get close to target count

        String updateExpression = (String) expressionData.get("updateExpression");
        @SuppressWarnings("unchecked")
        Map<String, AttributeValue> expressionAttributeValues =
            (Map<String, AttributeValue>)
expressionData.get("expressionAttributeValues");
        int actualCount = (int) expressionData.get("operatorCount");

        System.out.println("Generated update expression with approximately " +
actualCount + " operators");

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression(updateExpression)
            .expressionAttributeValues(expressionAttributeValues)
            .returnValues("UPDATED_NEW")
            .build();

        try {
            // Attempt the update operation
            UpdateItemResponse response = dynamoDbClient.updateItem(request);

            Map<String, Object> result = new HashMap<>();
            result.put("success", true);
```

```
            result.put("message", "Operation succeeded with " + actualCount + "
operators");
            result.put("data", response);
            return result;

        } catch (DynamoDbException e) {
            // Check if the error is due to exceeding the operator limit
            if (e.getMessage().contains("too many operators")) {
                Map<String, Object> result = new HashMap<>();
                result.put("success", false);
                result.put("message", "Operation failed: " + e.getMessage());
                result.put("operatorCount", actualCount);
                return result;
            }

            // Return other errors
            Map<String, Object> result = new HashMap<>();
            result.put("success", false);
            result.put("message", "Operation failed: " + e.getMessage());
            result.put("error", e);
            return result;
        }
    }

    /**
     * Break down a complex expression into multiple simpler operations.
     *
     * <p>This method demonstrates how to handle expressions that would exceed
     * the 300 operator limit by breaking them into multiple operations.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param totalOperations Total number of operations to perform
     * @return Map containing the results of the operations
     */
    public static Map<String, Object> breakDownComplexExpression(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, int totalOperations) {

        // Calculate how many operations we can safely include in each batch
        // Using 150 as a conservative limit (well below 300)
        final int operationsPerBatch = 100;
```

```
        final int batchCount = (int) Math.ceil((double) totalOperations /
operationsPerBatch);

        System.out.println("Breaking down " + totalOperations + " operations into "
+ batchCount + " batches");

        Map<String, Object> results = new HashMap<>();
        results.put("totalBatches", batchCount);

        Map<Integer, Map<String, Object>> batchResults = new HashMap<>();

        // Process each batch
        for (int batch = 0; batch < batchCount; batch++) {
            // Calculate the operations for this batch
            int batchStart = batch * operationsPerBatch;
            int batchEnd = Math.min(batchStart + operationsPerBatch,
totalOperations);
            int batchSize = batchEnd - batchStart;

            System.out.println(
                "Processing batch " + (batch + 1) + "/" + batchCount + " with " +
batchSize + " operations");

            // Create an update expression for this batch
            Map<String, Object> expressionData =
createComplexUpdateExpression(batchSize);

            String updateExpression = (String)
expressionData.get("updateExpression");
            @SuppressWarnings("unchecked")
            Map<String, AttributeValue> expressionAttributeValues =
                (Map<String, AttributeValue>)
expressionData.get("expressionAttributeValues");
            int operatorCount = (int) expressionData.get("operatorCount");

            // Define the update parameters
            UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression(updateExpression)
                .expressionAttributeValues(expressionAttributeValues)
                .returnValues("UPDATED_NEW")
                .build();
```

```
            try {
                // Perform the update operation for this batch
                UpdateItemResponse response = dynamoDbClient.updateItem(request);

                Map<String, Object> batchResult = new HashMap<>();
                batchResult.put("batch", batch + 1);
                batchResult.put("success", true);
                batchResult.put("operatorCount", operatorCount);
                batchResult.put("attributes", response.attributes());

                batchResults.put(batch, batchResult);

            } catch (DynamoDbException e) {
                Map<String, Object> batchResult = new HashMap<>();
                batchResult.put("batch", batch + 1);
                batchResult.put("success", false);
                batchResult.put("operatorCount", operatorCount);
                batchResult.put("error", e.getMessage());

                batchResults.put(batch, batchResult);

                // Continue with next batch instead of breaking
                continue;
            }
        }

        results.put("results", batchResults);
        return results;
    }

    /**
     * Count operators in a DynamoDB expression based on the rules in the
documentation.
     *
     * <p>This method demonstrates how operators are counted according to the
     * DynamoDB documentation.
     *
     * @param expression The DynamoDB expression to analyze
     * @return Map containing the breakdown of operator counts
     */
    public static Map<String, Integer> countOperatorsInExpression(String expression)
{
        // Initialize counters for different operator types
        Map<String, Integer> counts = new HashMap<>();
```

```
        counts.put("comparisonOperators", 0);
        counts.put("logicalOperators", 0);
        counts.put("functions", 0);
        counts.put("arithmeticOperators", 0);
        counts.put("specialOperators", 0);
        counts.put("total", 0);

        // Count comparison operators (=, <>, <, <=, >, >=)
        // This is a simplified approach and may not catch all cases
        int comparisonCount = 0;
        Pattern comparisonPattern = Pattern.compile("(=|<>|<=|>=|<|>)");
        Matcher comparisonMatcher = comparisonPattern.matcher(expression);
        while (comparisonMatcher.find()) {
            comparisonCount++;
        }
        counts.put("comparisonOperators", comparisonCount);

        // Count logical operators (AND, OR, NOT)
        int andCount = countOccurrences(expression, "\\bAND\\b");
        int orCount = countOccurrences(expression, "\\bOR\\b");
        int notCount = countOccurrences(expression, "\\bNOT\\b");
        counts.put("logicalOperators", andCount + orCount + notCount);

        // Count functions (attribute_exists, attribute_not_exists, attribute_type,
 begins_with, contains, size)
        int functionCount = countOccurrences(
            expression,
            "\\b(attribute_exists|attribute_not_exists|attribute_type|begins_with|
contains|size|if_not_exists)\\(");
        counts.put("functions", functionCount);

        // Count arithmetic operators (+ and -)
        // This is a simplified approach and may not catch all cases
        int arithmeticCount = 0;
        Pattern arithmeticPattern = Pattern.compile("[a-zA-Z0-9_)\\]]\\s*[\\+\\-]\
\s*[a-zA-Z0-9_:(]");
        Matcher arithmeticMatcher = arithmeticPattern.matcher(expression);
        while (arithmeticMatcher.find()) {
            arithmeticCount++;
        }
        counts.put("arithmeticOperators", arithmeticCount);

        // Count special operators (BETWEEN, IN)
        int betweenCount = countOccurrences(expression, "\\bBETWEEN\\b");
```

```
        int inCount = countOccurrences(expression, "\\bIN\\b");
        counts.put("specialOperators", betweenCount + inCount);

        // Add extra operators for BETWEEN (each BETWEEN includes an AND)
        int currentLogicalOps = counts.getOrDefault("logicalOperators", 0);
        counts.put("logicalOperators", currentLogicalOps + betweenCount);

        // Calculate total
        int total = counts.getOrDefault("comparisonOperators", 0)
            + counts.getOrDefault("logicalOperators", 0)
            + counts.getOrDefault("functions", 0)
            + counts.getOrDefault("arithmeticOperators", 0)
            + counts.getOrDefault("specialOperators", 0);
        counts.put("total", total);

        return counts;
    }

    /**
     * Helper method to count occurrences of a pattern in a string.
     *
     * @param text The text to search in
     * @param regex The regular expression pattern to search for
     * @return The number of occurrences
     */
    private static int countOccurrences(String text, String regex) {
        final Pattern pattern = Pattern.compile(regex);
        final Matcher matcher = pattern.matcher(text);
        int count = 0;
        while (matcher.find()) {
            count++;
        }
        return count;
    }
```

에서 계산하는 표현식 연산자의 사용 예입니다 AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());
```

```
        System.out.println("Demonstrating DynamoDB expression operator counting and
the 300 operator limit");

        try {
            // Example 1: Analyze a simple expression
            System.out.println("\nExample 1: Analyzing a simple expression");
            String simpleExpression = "Price = :price AND Rating > :rating AND
Category IN (:cat1, :cat2, :cat3)";
            Map<String, Integer> simpleCount =
countOperatorsInExpression(simpleExpression);

            System.out.println("Expression: " + simpleExpression);
            System.out.println("Operator count breakdown:");
            System.out.println("- Comparison operators: " +
simpleCount.get("comparisonOperators"));
            System.out.println("- Logical operators: " +
simpleCount.get("logicalOperators"));
            System.out.println("- Functions: " + simpleCount.get("functions"));
            System.out.println("- Arithmetic operators: " +
simpleCount.get("arithmeticOperators"));
            System.out.println("- Special operators: " +
simpleCount.get("specialOperators"));
            System.out.println("- Total operators: " + simpleCount.get("total"));

            // Example 2: Analyze a complex expression
            System.out.println("\nExample 2: Analyzing a complex expression");
            String complexExpression = "(attribute_exists(Category) AND Size
BETWEEN :min AND :max) OR "
                + "(Price > :price AND contains(Description, :keyword) AND "
                + "(Rating >= :minRating OR Reviews > :minReviews))";
            Map<String, Integer> complexCount =
countOperatorsInExpression(complexExpression);

            System.out.println("Expression: " + complexExpression);
            System.out.println("Operator count breakdown:");
            System.out.println("- Comparison operators: " +
complexCount.get("comparisonOperators"));
            System.out.println("- Logical operators: " +
complexCount.get("logicalOperators"));
            System.out.println("- Functions: " + complexCount.get("functions"));
            System.out.println("- Arithmetic operators: " +
complexCount.get("arithmeticOperators"));
```

```
            System.out.println("- Special operators: " +
complexCount.get("specialOperators"));
            System.out.println("- Total operators: " + complexCount.get("total"));

            // Example 3: Test approaching the operator limit
            System.out.println("\nExample 3: Testing an expression approaching the
operator limit");
            Map<String, Object> approachingLimit = testOperatorLimit(dynamoDbClient,
tableName, key, 290);
            System.out.println(approachingLimit.get("message"));

            // Example 4: Test exceeding the operator limit
            System.out.println("\nExample 4: Testing an expression exceeding the
operator limit");
            Map<String, Object> exceedingLimit = testOperatorLimit(dynamoDbClient,
tableName, key, 310);
            System.out.println(exceedingLimit.get("message"));

            // Example 5: Breaking down a complex expression
            System.out.println("\nExample 5: Breaking down a complex expression into
multiple operations");
            Map<String, Object> breakdownResult =
breakDownComplexExpression(dynamoDbClient, tableName, key, 500);
            @SuppressWarnings("unchecked")
            Map<Integer, Map<String, Object>> results =
                (Map<Integer, Map<String, Object>>) breakdownResult.get("results");
            System.out.println(
                "Processed " + results.size() + " of " +
breakdownResult.get("totalBatches") + " batches");

            // Explain the operator counting rules
            System.out.println("\nKey points about DynamoDB expression operator
counting:");
            System.out.println("1. The maximum number of operators in any expression
is 300");
            System.out.println("2. Each comparison operator (=, <>, <, <=, >, >=)
counts as 1 operator");
            System.out.println("3. Each logical operator (AND, OR, NOT) counts as 1
operator");
            System.out.println("4. Each function call (attribute_exists, contains,
etc.) counts as 1 operator");
            System.out.println("5. Each arithmetic operator (+ or -) counts as 1
operator");
```

```
            System.out.println("6. BETWEEN counts as 2 operators (BETWEEN itself and
the AND within it)");
            System.out.println("7. IN counts as 1 operator regardless of the number
of values");
            System.out.println("8. Parentheses for grouping and attribute paths
don't count as operators");
            System.out.println("9. When you exceed the limit, the error always
reports '301 operators'");
            System.out.println("10. For complex operations, break them into multiple
smaller operations");

        } catch (Exception e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateItem을 참조하세요.

사진을 관리하기 위한 서버리스 애플리케이션 만들기

다음 코드 예시에서는 사용자가 레이블을 사용하여 사진을 관리할 수 있는 서버리스 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 AWS 커뮤니티의 게시물을 참조하세요.

이 예제에서 사용되는 서비스

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

- Amazon SNS

글로벌 보조 인덱스가 있는 테이블 만들기

다음 코드 예제는 글로벌 보조 인덱스로 테이블을 만드는 방법을 보여줍니다.

SDK for Java 2.x

를 사용하여 글로벌 보조 인덱스가 있는 DynamoDB 테이블을 생성합니다 AWS SDK for Java 2.x.

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
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.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DeleteTableRequest;
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.GlobalSecondaryIndex;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.Projection;
import software.amazon.awssdk.services.dynamodb.model.ProjectionType;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    public void createTable() {
        try {
            // Attribute definitions
            final List<AttributeDefinition> attributeDefinitions = new
 ArrayList<>();
```

```
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName(ISSUE_ID_ATTR)
            .attributeType(ScalarAttributeType.S)
            .build());
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName(TITLE_ATTR)
            .attributeType(ScalarAttributeType.S)
            .build());
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName(CREATE_DATE_ATTR)
            .attributeType(ScalarAttributeType.S)
            .build());
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName(DUE_DATE_ATTR)
            .attributeType(ScalarAttributeType.S)
            .build());

        // Key schema for table
        final List<KeySchemaElement> tableKeySchema = new ArrayList<>();
        tableKeySchema.add(KeySchemaElement.builder()
            .attributeName(ISSUE_ID_ATTR)
            .keyType(KeyType.HASH)
            .build()); // Partition key
        tableKeySchema.add(KeySchemaElement.builder()
            .attributeName(TITLE_ATTR)
            .keyType(KeyType.RANGE)
            .build()); // Sort key

        // Initial provisioned throughput settings for the indexes
        final ProvisionedThroughput ptIndex = ProvisionedThroughput.builder()
            .readCapacityUnits(1L)
            .writeCapacityUnits(1L)
            .build();

        // CreateDateIndex
        final List<KeySchemaElement> createDateKeySchema = new ArrayList<>();
        createDateKeySchema.add(KeySchemaElement.builder()
            .attributeName(CREATE_DATE_ATTR)
            .keyType(KeyType.HASH)
            .build());
        createDateKeySchema.add(KeySchemaElement.builder()
            .attributeName(ISSUE_ID_ATTR)
            .keyType(KeyType.RANGE)
            .build());
```

```
            final Projection createDateProjection = Projection.builder()
                .projectionType(ProjectionType.INCLUDE)
                .nonKeyAttributes(DESCRIPTION_ATTR, STATUS_ATTR)
                .build();

            final GlobalSecondaryIndex createDateIndex =
GlobalSecondaryIndex.builder()
                .indexName(CREATE_DATE_INDEX)
                .keySchema(createDateKeySchema)
                .projection(createDateProjection)
                .provisionedThroughput(ptIndex)
                .build();

            // TitleIndex
            final List<KeySchemaElement> titleKeySchema = new ArrayList<>();
            titleKeySchema.add(KeySchemaElement.builder()
                .attributeName(TITLE_ATTR)
                .keyType(KeyType.HASH)
                .build());
            titleKeySchema.add(KeySchemaElement.builder()
                .attributeName(ISSUE_ID_ATTR)
                .keyType(KeyType.RANGE)
                .build());

            final Projection titleProjection =

Projection.builder().projectionType(ProjectionType.KEYS_ONLY).build();

            final GlobalSecondaryIndex titleIndex = GlobalSecondaryIndex.builder()
                .indexName(TITLE_INDEX)
                .keySchema(titleKeySchema)
                .projection(titleProjection)
                .provisionedThroughput(ptIndex)
                .build();

            // DueDateIndex
            final List<KeySchemaElement> dueDateKeySchema = new ArrayList<>();
            dueDateKeySchema.add(KeySchemaElement.builder()
                .attributeName(DUE_DATE_ATTR)
                .keyType(KeyType.HASH)
                .build());

            final Projection dueDateProjection =
```

```
                Projection.builder().projectionType(ProjectionType.ALL).build();

            final GlobalSecondaryIndex dueDateIndex = GlobalSecondaryIndex.builder()
                .indexName(DUE_DATE_INDEX)
                .keySchema(dueDateKeySchema)
                .projection(dueDateProjection)
                .provisionedThroughput(ptIndex)
                .build();

            final CreateTableRequest createTableRequest =
CreateTableRequest.builder()
                .tableName(TABLE_NAME)
                .keySchema(tableKeySchema)
                .attributeDefinitions(attributeDefinitions)
                .globalSecondaryIndexes(createDateIndex, titleIndex, dueDateIndex)
                .provisionedThroughput(ProvisionedThroughput.builder()
                    .readCapacityUnits(1L)
                    .writeCapacityUnits(1L)
                    .build())
                .build();

            System.out.println("Creating table " + TABLE_NAME + "...");
            dynamoDbClient.createTable(createTableRequest);

            // Wait for table to become active
            System.out.println("Waiting for " + TABLE_NAME + " to become
ACTIVE...");
            final DynamoDbWaiter waiter = dynamoDbClient.waiter();
            final DescribeTableRequest describeTableRequest =
                DescribeTableRequest.builder().tableName(TABLE_NAME).build();

            final WaiterResponse<DescribeTableResponse> waiterResponse =
                waiter.waitUntilTableExists(describeTableRequest);
            waiterResponse.matched().response().ifPresent(response ->
System.out.println("Table is now ready for use"));

        } catch (DynamoDbException e) {
            System.err.println("Error creating table: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateTable을 참조하세요.

웜 처리량이 활성화된 테이블 만들기

다음 코드 예제에서는 웜 처리량이 활성화된 테이블을 만드는 방법을 보여줍니다.

SDK for Java 2.x

AWS SDK for Java 2.x를 사용하여 웜 처리량 설정이 있는 DynamoDB 테이블을 만듭니다.

```
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.GlobalSecondaryIndex;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.Projection;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.model.WarmThroughput;

    public static WarmThroughput buildWarmThroughput(final Long readUnitsPerSecond,
 final Long writeUnitsPerSecond) {
        return WarmThroughput.builder()
            .readUnitsPerSecond(readUnitsPerSecond)
            .writeUnitsPerSecond(writeUnitsPerSecond)
            .build();
    }

    /**
     * Builds a ProvisionedThroughput object with the specified read and write
 capacity units.
     *
     * @param readCapacityUnits The read capacity units
     * @param writeCapacityUnits The write capacity units
     * @return A configured ProvisionedThroughput object
     */
    public static ProvisionedThroughput buildProvisionedThroughput(
        final Long readCapacityUnits, final Long writeCapacityUnits) {
        return ProvisionedThroughput.builder()
            .readCapacityUnits(readCapacityUnits)
            .writeCapacityUnits(writeCapacityUnits)
            .build();
    }
```

```
    /**
     * Builds an AttributeDefinition with the specified name and type.
     *
     * @param attributeName The attribute name
     * @param scalarAttributeType The attribute type
     * @return A configured AttributeDefinition
     */
    private static AttributeDefinition buildAttributeDefinition(
        final String attributeName, final ScalarAttributeType scalarAttributeType) {
        return AttributeDefinition.builder()
            .attributeName(attributeName)
            .attributeType(scalarAttributeType)
            .build();
    }

    /**
     * Builds a KeySchemaElement with the specified name and key type.
     *
     * @param attributeName The attribute name
     * @param keyType The key type (HASH or RANGE)
     * @return A configured KeySchemaElement
     */
    private static KeySchemaElement buildKeySchemaElement(final String
attributeName, final KeyType keyType) {
        return KeySchemaElement.builder()
            .attributeName(attributeName)
            .keyType(keyType)
            .build();
    }

    /**
     * Creates a DynamoDB table with the specified configuration including warm
throughput settings.
     *
     * @param ddb The DynamoDB client
     * @param tableName The name of the table to create
     * @param partitionKey The partition key attribute name
     * @param sortKey The sort key attribute name
     * @param miscellaneousKeyAttribute Additional key attribute name for GSI
     * @param nonKeyAttribute Non-key attribute to include in GSI projection
     * @param tableReadCapacityUnits Read capacity units for the table
     * @param tableWriteCapacityUnits Write capacity units for the table
     * @param tableWarmReadUnitsPerSecond Warm read units per second for the table
     * @param tableWarmWriteUnitsPerSecond Warm write units per second for the table
```

```
    * @param globalSecondaryIndexName The name of the GSI to create
    * @param globalSecondaryIndexReadCapacityUnits Read capacity units for the GSI
    * @param globalSecondaryIndexWriteCapacityUnits Write capacity units for the
GSI
    * @param globalSecondaryIndexWarmReadUnitsPerSecond Warm read units per second
for the GSI
    * @param globalSecondaryIndexWarmWriteUnitsPerSecond Warm write units per
second for the GSI
    */
   public static void createDynamoDBTable(
       final DynamoDbClient ddb,
       final String tableName,
       final String partitionKey,
       final String sortKey,
       final String miscellaneousKeyAttribute,
       final String nonKeyAttribute,
       final Long tableReadCapacityUnits,
       final Long tableWriteCapacityUnits,
       final Long tableWarmReadUnitsPerSecond,
       final Long tableWarmWriteUnitsPerSecond,
       final String globalSecondaryIndexName,
       final Long globalSecondaryIndexReadCapacityUnits,
       final Long globalSecondaryIndexWriteCapacityUnits,
       final Long globalSecondaryIndexWarmReadUnitsPerSecond,
       final Long globalSecondaryIndexWarmWriteUnitsPerSecond) {

       // Define the table attributes
       final AttributeDefinition partitionKeyAttribute =
buildAttributeDefinition(partitionKey, ScalarAttributeType.S);
       final AttributeDefinition sortKeyAttribute =
buildAttributeDefinition(sortKey, ScalarAttributeType.S);
       final AttributeDefinition miscellaneousKeyAttributeDefinition =
           buildAttributeDefinition(miscellaneousKeyAttribute,
ScalarAttributeType.N);
       final AttributeDefinition[] attributeDefinitions = {
           partitionKeyAttribute, sortKeyAttribute,
miscellaneousKeyAttributeDefinition
       };

       // Define the table key schema
       final KeySchemaElement partitionKeyElement =
buildKeySchemaElement(partitionKey, KeyType.HASH);
       final KeySchemaElement sortKeyElement = buildKeySchemaElement(sortKey,
KeyType.RANGE);
```

```
        final KeySchemaElement[] keySchema = {partitionKeyElement, sortKeyElement};

        // Define the provisioned throughput for the table
        final ProvisionedThroughput provisionedThroughput =
            buildProvisionedThroughput(tableReadCapacityUnits,
tableWriteCapacityUnits);

        // Define the Global Secondary Index (GSI)
        final KeySchemaElement globalSecondaryIndexPartitionKeyElement =
buildKeySchemaElement(sortKey, KeyType.HASH);
        final KeySchemaElement globalSecondaryIndexSortKeyElement =
            buildKeySchemaElement(miscellaneousKeyAttribute, KeyType.RANGE);
        final KeySchemaElement[] gsiKeySchema = {
            globalSecondaryIndexPartitionKeyElement,
globalSecondaryIndexSortKeyElement
        };

        final Projection gsiProjection = Projection.builder()
            .projectionType(PROJECTION_TYPE_INCLUDE)
            .nonKeyAttributes(nonKeyAttribute)
            .build();

        final ProvisionedThroughput gsiProvisionedThroughput =
            buildProvisionedThroughput(globalSecondaryIndexReadCapacityUnits,
globalSecondaryIndexWriteCapacityUnits);

        // Define the warm throughput for the Global Secondary Index (GSI)
        final WarmThroughput gsiWarmThroughput = buildWarmThroughput(
            globalSecondaryIndexWarmReadUnitsPerSecond,
globalSecondaryIndexWarmWriteUnitsPerSecond);

        final GlobalSecondaryIndex globalSecondaryIndex =
GlobalSecondaryIndex.builder()
            .indexName(globalSecondaryIndexName)
            .keySchema(gsiKeySchema)
            .projection(gsiProjection)
            .provisionedThroughput(gsiProvisionedThroughput)
            .warmThroughput(gsiWarmThroughput)
            .build();

        // Define the warm throughput for the table
        final WarmThroughput tableWarmThroughput =
            buildWarmThroughput(tableWarmReadUnitsPerSecond,
tableWarmWriteUnitsPerSecond);
```

```
        final CreateTableRequest request = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(attributeDefinitions)
            .keySchema(keySchema)
            .provisionedThroughput(provisionedThroughput)
            .globalSecondaryIndexes(globalSecondaryIndex)
            .warmThroughput(tableWarmThroughput)
            .build();

        final CreateTableResponse response = ddb.createTable(request);
        System.out.println(response);
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateTable을 참조하세요.

DynamoDB 데이터를 추적하는 웹 애플리케이션 만들기

다음 코드 예제에서는 Amazon DynamoDB 테이블에서 작업 항목을 추적하고 Amazon Simple Email Service(Amazon SES)를 사용하여 보고서를 보내는 웹 애플리케이션을 만드는 방법을 보여줍니다.

SDK for Java 2.x

Amazon DynamoDB API를 사용하여 DynamoDB 작업 데이터를 추적하는 동적 웹 애플리케이션을 만드는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- DynamoDB
- Amazon SES

TTL을 사용하여 항목 생성

다음 코드 예제에서는 TTL을 사용해 항목을 만드는 방법을 보여줍니다.

SDK for Java 2.x

```
package com.amazon.samplelib.ttl;
```

```
import com.amazon.samplelib.CodeSampleUtils;
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;
import java.util.Map;
import java.util.Optional;

/**
 * Creates an item in a DynamoDB table with TTL attributes.
 * This class demonstrates how to add TTL expiration timestamps to DynamoDB items.
 */
public class CreateTTL {

    private static 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)
            """;
    private static final int DAYS_TO_EXPIRE = 90;
    private static final int SECONDS_PER_DAY = 24 * 60 * 60;
    private static final String PRIMARY_KEY_ATTR = "primaryKey";
    private static final String SORT_KEY_ATTR = "sortKey";
    private static final String CREATION_DATE_ATTR = "creationDate";
    private static final String EXPIRE_AT_ATTR = "expireAt";
    private static final String SUCCESS_MESSAGE = "%s PutItem operation with TTL
 successful.";
    private static final String TABLE_NOT_FOUND_ERROR = "Error: The Amazon DynamoDB
 table \"%s\" can't be found.";
```

```
    private final DynamoDbClient dynamoDbClient;

    /**
     * Constructs a CreateTTL instance with the specified DynamoDB client.
     *
     * @param dynamoDbClient The DynamoDB client to use
     */
    public CreateTTL(final DynamoDbClient dynamoDbClient) {
        this.dynamoDbClient = dynamoDbClient;
    }

    /**
     * Constructs a CreateTTL with a default DynamoDB client.
     */
    public CreateTTL() {
        this.dynamoDbClient = null;
    }

    /**
     * Main method to demonstrate creating an item with TTL.
     *
     * @param args Command line arguments
     */
    public static void main(final String[] args) {
        try {
            int result = new CreateTTL().processArgs(args);
            System.exit(result);
        } catch (Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Process command line arguments and create an item with TTL.
     *
     * @param args Command line arguments
     * @return 0 if successful, non-zero otherwise
     * @throws ResourceNotFoundException If the table doesn't exist
     * @throws DynamoDbException If an error occurs during the operation
     * @throws IllegalArgumentException If arguments are invalid
     */
    public int processArgs(final String[] args) {
        // Argument validation (remove or replace this line when reusing this code)
```

```
        CodeSampleUtils.validateArgs(args, new int[] {3, 4}, USAGE);

        final String tableName = args[0];
        final String primaryKey = args[1];
        final String sortKey = args[2];
        final Region region = Optional.ofNullable(args.length > 3 ? args[3] : null)
            .map(Region::of)
            .orElse(Region.US_EAST_1);

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            final CreateTTL createTTL = new CreateTTL(ddb);
            createTTL.createItemWithTTL(tableName, primaryKey, sortKey);
            return 0;
        } catch (Exception e) {
            throw e;
        }
    }

    /**
     * Creates an item in the specified table with TTL attributes.
     *
     * @param tableName The name of the table
     * @param primaryKeyValue The value for the primary key
     * @param sortKeyValue The value for the sort key
     * @return The response from the PutItem operation
     * @throws ResourceNotFoundException If the table doesn't exist
     * @throws DynamoDbException If an error occurs during the operation
     */
    public PutItemResponse createItemWithTTL(
        final String tableName, final String primaryKeyValue, final String
sortKeyValue) {
        // 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 + (DAYS_TO_EXPIRE * SECONDS_PER_DAY);

        final Map<String, AttributeValue> itemMap = new HashMap<>();
        itemMap.put(
            PRIMARY_KEY_ATTR, AttributeValue.builder().s(primaryKeyValue).build());
        itemMap.put(SORT_KEY_ATTR,
AttributeValue.builder().s(sortKeyValue).build());
```

```
        itemMap.put(
            CREATION_DATE_ATTR,
            AttributeValue.builder().n(String.valueOf(createDate)).build());
        itemMap.put(
            EXPIRE_AT_ATTR,
            AttributeValue.builder().n(String.valueOf(expireDate)).build());

        final PutItemRequest request =
            PutItemRequest.builder().tableName(tableName).item(itemMap).build();

        try {
            final PutItemResponse response = dynamoDbClient.putItem(request);
            System.out.println(String.format(SUCCESS_MESSAGE, tableName));
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutItem을 참조하세요.

MRSC 글로벌 테이블 만들기 및 관리

다음 코드 예제에서는 다중 리전 강력한 일관성(MRSC)을 사용하여 DynamoDB 글로벌 테이블을 만들고 관리하는 방법을 보여줍니다.

- 다중 리전 강력한 일관성이 있는 테이블을 만듭니다.
- MRSC 구성 및 복제본 상태를 확인합니다.
- 즉각적인 읽기를 통해 리전 간 강력한 일관성을 테스트합니다.
- MRSC 보장을 사용하여 조건부 쓰기를 수행합니다.
- MRSC 글로벌 테이블 리소스를 정리합니다.

SDK for Java 2.x

를 사용하여 MRSC 변환에 사용할 수 있는 리전 테이블을 생성합니다 AWS SDK for Java 2.x.

```
   public static CreateTableResponse createRegionalTable(final DynamoDbClient
dynamoDbClient, final String tableName) {

       if (dynamoDbClient == null) {
           throw new IllegalArgumentException("DynamoDB client cannot be null");
       }
       if (tableName == null || tableName.trim().isEmpty()) {
           throw new IllegalArgumentException("Table name cannot be null or
empty");
       }

       try {
           LOGGER.info("Creating regional table: " + tableName + " (must be empty
for MRSC)");

           CreateTableRequest createTableRequest = CreateTableRequest.builder()
               .tableName(tableName)
               .attributeDefinitions(
                   AttributeDefinition.builder()
                       .attributeName("Artist")
                       .attributeType(ScalarAttributeType.S)
                       .build(),
                   AttributeDefinition.builder()
                       .attributeName("SongTitle")
                       .attributeType(ScalarAttributeType.S)
                       .build())
               .keySchema(
                   KeySchemaElement.builder()
                       .attributeName("Artist")
                       .keyType(KeyType.HASH)
                       .build(),
                   KeySchemaElement.builder()
                       .attributeName("SongTitle")
                       .keyType(KeyType.RANGE)
                       .build())
               .billingMode(BillingMode.PAY_PER_REQUEST)
               .build();

           CreateTableResponse response =
dynamoDbClient.createTable(createTableRequest);
```

```
            LOGGER.info("Regional table creation initiated. Status: "
                + response.tableDescription().tableStatus());

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to create regional table: " + tableName + " - " +
e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to create regional table: " + tableName)
                .cause(e)
                .build();
        }
    }
```

를 사용하여 복제본 및 감시를 사용하여 리전 테이블을 MRSC로 변환합니다 AWS SDK for Java 2.x.

```
    public static UpdateTableResponse convertToMRSCWithWitness(
        final DynamoDbClient dynamoDbClient,
        final String tableName,
        final Region replicaRegion,
        final Region witnessRegion) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (replicaRegion == null) {
            throw new IllegalArgumentException("Replica region cannot be null");
        }
        if (witnessRegion == null) {
            throw new IllegalArgumentException("Witness region cannot be null");
        }

        try {
            LOGGER.info("Converting table to MRSC with replica in " +
replicaRegion.id() + " and witness in "
                + witnessRegion.id());
```

```
            // Create replica update using ReplicationGroupUpdate
            ReplicationGroupUpdate replicaUpdate = ReplicationGroupUpdate.builder()
                .create(CreateReplicationGroupMemberAction.builder()
                    .regionName(replicaRegion.id())
                    .build())
                .build();

            // Create witness update
            GlobalTableWitnessGroupUpdate witnessUpdate =
GlobalTableWitnessGroupUpdate.builder()
                .create(CreateGlobalTableWitnessGroupMemberAction.builder()
                    .regionName(witnessRegion.id())
                    .build())
                .build();

            UpdateTableRequest updateTableRequest = UpdateTableRequest.builder()
                .tableName(tableName)
                .replicaUpdates(List.of(replicaUpdate))
                .globalTableWitnessUpdates(List.of(witnessUpdate))
                .multiRegionConsistency(MultiRegionConsistency.STRONG)
                .build();

            UpdateTableResponse response =
dynamoDbClient.updateTable(updateTableRequest);
            LOGGER.info("MRSC conversion initiated. Status: "
                + response.tableDescription().tableStatus());
            LOGGER.info("UpdateTableResponse full object: " + response);
            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to convert table to MRSC: " + tableName + " - " +
e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to convert table to MRSC: " + tableName)
                .cause(e)
                .build();
        }
    }
```

를 사용하여 MRSC 글로벌 테이블 구성을 설명합니다 AWS SDK for Java 2.x.

```
   public static DescribeTableResponse describeMRSCTable(final DynamoDbClient
dynamoDbClient, final String tableName) {

       if (dynamoDbClient == null) {
           throw new IllegalArgumentException("DynamoDB client cannot be null");
       }
       if (tableName == null || tableName.trim().isEmpty()) {
           throw new IllegalArgumentException("Table name cannot be null or
empty");
       }

       try {
           LOGGER.info("Describing MRSC global table: " + tableName);

           DescribeTableRequest request =
               DescribeTableRequest.builder().tableName(tableName).build();

           DescribeTableResponse response = dynamoDbClient.describeTable(request);

           LOGGER.info("Table status: " + response.table().tableStatus());
           LOGGER.info("Multi-region consistency: " +
response.table().multiRegionConsistency());

           if (response.table().replicas() != null
               && !response.table().replicas().isEmpty()) {
               LOGGER.info("Number of replicas: " +
response.table().replicas().size());
               response.table()
                   .replicas()
                   .forEach(replica -> LOGGER.info(
                       "Replica region: " + replica.regionName() + ", Status: " +
replica.replicaStatus()));
           }

           if (response.table().globalTableWitnesses() != null
               && !response.table().globalTableWitnesses().isEmpty()) {
               LOGGER.info("Number of witnesses: "
                   + response.table().globalTableWitnesses().size());
               response.table()
                   .globalTableWitnesses()
                   .forEach(witness -> LOGGER.info(
                       "Witness region: " + witness.regionName() + ", Status: " +
witness.witnessStatus()));
```

```
            }

            return response;

        } catch (ResourceNotFoundException e) {
            LOGGER.severe("Table not found: " + tableName + " - " + e.getMessage());
            throw DynamoDbException.builder()
                .message("Table not found: " + tableName)
                .cause(e)
                .build();
        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to describe table: " + tableName + " - " +
e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to describe table: " + tableName)
                .cause(e)
                .build();
        }
    }
```

AWS SDK for Java 2.x를 사용하여 MRSC 강력한 일관성을 확인하기 위한 테스트 항목을 추가합니다.

```
    public static PutItemResponse putTestItem(
        final DynamoDbClient dynamoDbClient,
        final String tableName,
        final String artist,
        final String songTitle,
        final String album,
        final String year) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (artist == null || artist.trim().isEmpty()) {
            throw new IllegalArgumentException("Artist cannot be null or empty");
        }
        if (songTitle == null || songTitle.trim().isEmpty()) {
```

```
            throw new IllegalArgumentException("Song title cannot be null or
empty");
        }

        try {
            LOGGER.info("Adding test item to MRSC global table: " + tableName);

            Map<String, AttributeValue> item = new HashMap<>();
            item.put("Artist", AttributeValue.builder().s(artist).build());
            item.put("SongTitle", AttributeValue.builder().s(songTitle).build());

            if (album != null && !album.trim().isEmpty()) {
                item.put("Album", AttributeValue.builder().s(album).build());
            }
            if (year != null && !year.trim().isEmpty()) {
                item.put("Year", AttributeValue.builder().n(year).build());
            }

            PutItemRequest putItemRequest =
                PutItemRequest.builder().tableName(tableName).item(item).build();

            PutItemResponse response = dynamoDbClient.putItem(putItemRequest);
            LOGGER.info("Test item added successfully with strong consistency");

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to add test item to table: " + tableName + " - " +
e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to add test item to table: " + tableName)
                .cause(e)
                .build();
        }
    }
```

를 사용하여 MRSC 복제본에서 일관된 읽기로 항목을 읽습니다 AWS SDK for Java 2.x.

```
    public static GetItemResponse getItemWithConsistentRead(
        final DynamoDbClient dynamoDbClient, final String tableName, final String
artist, final String songTitle) {
```

```
        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (artist == null || artist.trim().isEmpty()) {
            throw new IllegalArgumentException("Artist cannot be null or empty");
        }
        if (songTitle == null || songTitle.trim().isEmpty()) {
            throw new IllegalArgumentException("Song title cannot be null or
empty");
        }

        try {
            LOGGER.info("Reading item from MRSC global table with consistent read: "
+ tableName);

            Map<String, AttributeValue> key = new HashMap<>();
            key.put("Artist", AttributeValue.builder().s(artist).build());
            key.put("SongTitle", AttributeValue.builder().s(songTitle).build());

            GetItemRequest getItemRequest = GetItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .consistentRead(true)
                .build();

            GetItemResponse response = dynamoDbClient.getItem(getItemRequest);

            if (response.hasItem()) {
                LOGGER.info("Item found with strong consistency - no wait time
needed");
            } else {
                LOGGER.info("Item not found");
            }

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to read item from table: " + tableName + " - " +
e.getMessage());
            throw DynamoDbException.builder()
```

```
                .message("Failed to read item from table: " + tableName)
                .cause(e)
                .build();
        }
    }
```

를 사용하여 MRSC 보장으로 조건부 업데이트를 수행합니다 AWS SDK for Java 2.x.

```
    public static UpdateItemResponse performConditionalUpdate(
        final DynamoDbClient dynamoDbClient,
        final String tableName,
        final String artist,
        final String songTitle,
        final String rating) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (artist == null || artist.trim().isEmpty()) {
            throw new IllegalArgumentException("Artist cannot be null or empty");
        }
        if (songTitle == null || songTitle.trim().isEmpty()) {
            throw new IllegalArgumentException("Song title cannot be null or
empty");
        }
        if (rating == null || rating.trim().isEmpty()) {
            throw new IllegalArgumentException("Rating cannot be null or empty");
        }

        try {
            LOGGER.info("Performing conditional update on MRSC global table: " +
tableName);

            Map<String, AttributeValue> key = new HashMap<>();
            key.put("Artist", AttributeValue.builder().s(artist).build());
            key.put("SongTitle", AttributeValue.builder().s(songTitle).build());

            Map<String, String> expressionAttributeNames = new HashMap<>();
```

```
            expressionAttributeNames.put("#rating", "Rating");

            Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
            expressionAttributeValues.put(
                ":rating", AttributeValue.builder().n(rating).build());

            UpdateItemRequest updateItemRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #rating = :rating")
                .conditionExpression("attribute_exists(Artist)")
                .expressionAttributeNames(expressionAttributeNames)
                .expressionAttributeValues(expressionAttributeValues)
                .build();

            UpdateItemResponse response =
dynamoDbClient.updateItem(updateItemRequest);
            LOGGER.info("Conditional update successful - demonstrates strong
consistency");

            return response;

        } catch (ConditionalCheckFailedException e) {
            LOGGER.warning("Conditional check failed: " + e.getMessage());
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to perform conditional update: " + tableName + " -
" + e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to perform conditional update: " + tableName)
                .cause(e)
                .build();
        }
    }
```

를 사용하여 MRSC 복제본과 감시자가 활성화될 때까지 기다립니다 AWS SDK for Java 2.x.

```
    public static void waitForMRSCReplicasActive(
        final DynamoDbClient dynamoDbClient, final String tableName, final int
maxWaitTimeSeconds)
        throws InterruptedException {
```

```
        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (maxWaitTimeSeconds <= 0) {
            throw new IllegalArgumentException("Max wait time must be positive");
        }

        try {
            LOGGER.info("Waiting for MRSC replicas and witnesses to become active: "
+ tableName);

            final long startTime = System.currentTimeMillis();
            final long maxWaitTimeMillis = maxWaitTimeSeconds * 1000L;
            int backoffSeconds = 5; // Start with 5 second intervals
            final int maxBackoffSeconds = 30; // Cap at 30 seconds

            while (System.currentTimeMillis() - startTime < maxWaitTimeMillis) {
                DescribeTableResponse response = describeMRSCTable(dynamoDbClient,
tableName);

                boolean allActive = true;
                StringBuilder statusReport = new StringBuilder();

                if (response.table().multiRegionConsistency() == null
                    || !MultiRegionConsistency.STRONG
                        .toString()

.equals(response.table().multiRegionConsistency().toString())) {
                    allActive = false;
                    statusReport
                        .append("MultiRegionConsistency: ")
                        .append(response.table().multiRegionConsistency())
                        .append(" ");
                }
                if (response.table().replicas() == null
                    || response.table().replicas().isEmpty()) {
                    allActive = false;
                    statusReport.append("No replicas found. ");
                }
                if (response.table().globalTableWitnesses() == null
```

```
                    || response.table().globalTableWitnesses().isEmpty()) {
                    allActive = false;
                    statusReport.append("No witnesses found. ");
                }

                // Check table status
                if (!"ACTIVE".equals(response.table().tableStatus().toString())) {
                    allActive = false;
                    statusReport
                        .append("Table: ")
                        .append(response.table().tableStatus())
                        .append(" ");
                }

                // Check replica status
                if (response.table().replicas() != null) {
                    for (var replica : response.table().replicas()) {
                        if (!"ACTIVE".equals(replica.replicaStatus().toString())) {
                            allActive = false;
                            statusReport
                                .append("Replica(")
                                .append(replica.regionName())
                                .append("): ")
                                .append(replica.replicaStatus())
                                .append(" ");
                        }
                    }
                }

                // Check witness status
                if (response.table().globalTableWitnesses() != null) {
                    for (var witness : response.table().globalTableWitnesses()) {
                        if (!"ACTIVE".equals(witness.witnessStatus().toString())) {
                            allActive = false;
                            statusReport
                                .append("Witness(")
                                .append(witness.regionName())
                                .append("): ")
                                .append(witness.witnessStatus())
                                .append(" ");
                        }
                    }
                }
```

```
                if (allActive) {
                    LOGGER.info("All MRSC replicas and witnesses are now active: " +
tableName);
                    return;
                }

                LOGGER.info("Waiting for MRSC components to become active. Status: "
+ statusReport.toString());
                LOGGER.info("Next check in " + backoffSeconds + " seconds...");

                tempWait(backoffSeconds);

                // Exponential backoff with cap
                backoffSeconds = Math.min(backoffSeconds * 2, maxBackoffSeconds);
            }

            throw DynamoDbException.builder()
                .message("Timeout waiting for MRSC replicas to become active after "
+ maxWaitTimeSeconds + " seconds")
                .build();

        } catch (DynamoDbException | InterruptedException e) {
            LOGGER.severe("Failed to wait for MRSC replicas to become active: " +
tableName + " - " + e.getMessage());
            throw e;
        }
    }
```

를 사용하여 MRSC 복제본 및 감시자를 정리합니다 AWS SDK for Java 2.x.

```
    public static UpdateTableResponse cleanupMRSCReplicas(
        final DynamoDbClient dynamoDbClient,
        final String tableName,
        final Region replicaRegion,
        final Region witnessRegion) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
```

```
        }
        if (replicaRegion == null) {
            throw new IllegalArgumentException("Replica region cannot be null");
        }
        if (witnessRegion == null) {
            throw new IllegalArgumentException("Witness region cannot be null");
        }

        try {
            LOGGER.info("Cleaning up MRSC replicas and witnesses for table: " +
tableName);

            // Remove replica using ReplicationGroupUpdate
            ReplicationGroupUpdate replicaUpdate = ReplicationGroupUpdate.builder()
                .delete(DeleteReplicationGroupMemberAction.builder()
                    .regionName(replicaRegion.id())
                    .build())
                .build();

            // Remove witness
            GlobalTableWitnessGroupUpdate witnessUpdate =
GlobalTableWitnessGroupUpdate.builder()
                .delete(DeleteGlobalTableWitnessGroupMemberAction.builder()
                    .regionName(witnessRegion.id())
                    .build())
                .build();

            UpdateTableRequest updateTableRequest = UpdateTableRequest.builder()
                .tableName(tableName)
                .replicaUpdates(List.of(replicaUpdate))
                .globalTableWitnessUpdates(List.of(witnessUpdate))
                .build();

            UpdateTableResponse response =
dynamoDbClient.updateTable(updateTableRequest);
            LOGGER.info("MRSC cleanup initiated - removing replica and witness.
Response: " + response);

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to cleanup MRSC replicas: " + tableName + " - " +
e.getMessage());
            throw DynamoDbException.builder()
```

```
                .message("Failed to cleanup MRSC replicas: " + tableName)
                .cause(e)
                .build();
        }
    }
```

를 사용하여 MRSC 워크플로 데모를 완료합니다 AWS SDK for Java 2.x.

```
    public static void demonstrateCompleteMRSCWorkflow(
        final DynamoDbClient primaryClient,
        final DynamoDbClient replicaClient,
        final String tableName,
        final Region replicaRegion,
        final Region witnessRegion)
        throws InterruptedException {

        if (primaryClient == null) {
            throw new IllegalArgumentException("Primary DynamoDB client cannot be
null");
        }
        if (replicaClient == null) {
            throw new IllegalArgumentException("Replica DynamoDB client cannot be
null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (replicaRegion == null) {
            throw new IllegalArgumentException("Replica region cannot be null");
        }
        if (witnessRegion == null) {
            throw new IllegalArgumentException("Witness region cannot be null");
        }

        try {
            LOGGER.info("=== Starting Complete MRSC Workflow Demonstration ===");

            // Step 1: Create an empty single-Region table
            LOGGER.info("Step 1: Creating empty single-Region table");
            createRegionalTable(primaryClient, tableName);
```

```
            // Use the existing GlobalTableOperations method for basic table waiting
            LOGGER.info("Intermediate step: Waiting for table [" + tableName + "] to
become active before continuing");
            GlobalTableOperations.waitForTableActive(primaryClient, tableName);

            // Step 2: Convert to MRSC with replica and witness
            LOGGER.info("Step 2: Converting to MRSC with replica and witness");
            convertToMRSCWithWitness(primaryClient, tableName, replicaRegion,
witnessRegion);

            // Wait for MRSC conversion to complete using MRSC-specific waiter
            LOGGER.info("Waiting for MRSC conversion to complete...");
            waitForMRSCReplicasActive(primaryClient, tableName);

            LOGGER.info("Intermediate step: Waiting for table [" + tableName + "] to
become active before continuing");
            GlobalTableOperations.waitForTableActive(primaryClient, tableName);

            // Step 3: Verify MRSC configuration
            LOGGER.info("Step 3: Verifying MRSC configuration");
            describeMRSCTable(primaryClient, tableName);

            // Step 4: Test strong consistency with data operations
            LOGGER.info("Step 4: Testing strong consistency with data operations");

            // Add test item to primary region
            putTestItem(primaryClient, tableName, "The Beatles", "Hey Jude", "The
Beatles 1967-1970", "1968");

            // Immediately read from replica region (no wait needed with MRSC)
            LOGGER.info("Reading from replica region immediately (strong
consistency):");
            GetItemResponse getResponse =
                getItemWithConsistentRead(replicaClient, tableName, "The Beatles",
"Hey Jude");

            if (getResponse.hasItem()) {
                LOGGER.info("# Strong consistency verified - item immediately
available in replica region");
            } else {
                LOGGER.warning("# Item not found in replica region");
            }

            // Test conditional update from replica region
```

```
            LOGGER.info("Testing conditional update from replica region:");
            performConditionalUpdate(replicaClient, tableName, "The Beatles", "Hey
Jude", "5");
            LOGGER.info("# Conditional update successful - demonstrates strong
consistency");

            // Step 5: Cleanup
            LOGGER.info("Step 5: Cleaning up resources");
            cleanupMRSCReplicas(primaryClient, tableName, replicaRegion,
witnessRegion);

            // Wait for cleanup to complete using basic table waiter
            LOGGER.info("Waiting for replica cleanup to complete...");
            GlobalTableOperations.waitForTableActive(primaryClient, tableName);

            // "Halt" until replica/witness cleanup is complete
            DescribeTableResponse cleanupVerification =
describeMRSCTable(primaryClient, tableName);
            int backoffSeconds = 5; // Start with 5 second intervals
            while (cleanupVerification.table().multiRegionConsistency() != null) {
                LOGGER.info("Waiting additional time (" + backoffSeconds + "
seconds) for MRSC cleanup to complete...");
                tempWait(backoffSeconds);

                // Exponential backoff with cap
                backoffSeconds = Math.min(backoffSeconds * 2, 30);
                cleanupVerification = describeMRSCTable(primaryClient, tableName);
            }

            // Delete the primary table
            deleteTable(primaryClient, tableName);

            LOGGER.info("=== MRSC Workflow Demonstration Complete ===");
            LOGGER.info("");
            LOGGER.info("Key benefits of Multi-Region Strong Consistency (MRSC):");
            LOGGER.info("- Immediate consistency across all regions (no eventual
consistency delays)");
            LOGGER.info("- Simplified application logic (no need to handle eventual
consistency)");
            LOGGER.info("- Support for conditional writes and transactions across
regions");
            LOGGER.info("- Consistent read operations from any region without
waiting");
```

```
        } catch (DynamoDbException | InterruptedException e) {
            LOGGER.severe("MRSC workflow failed: " + e.getMessage());
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - CreateTable
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - UpdateItem
  - UpdateTable

MREC를 보여주는 글로벌 테이블 만들기 및 관리

다음 코드 예제에서는 다중 리전 간 복제본을 사용하여 DynamoDB 글로벌 테이블을 만들고 관리하는 방법을 보여줍니다.

- 글로벌 보조 인덱스와 DynamoDB Streams가 있는 테이블을 만듭니다.
- 다른 리전에 복제본을 추가하여 글로벌 테이블을 만듭니다.
- 글로벌 테이블에서 복제본을 제거합니다.
- 테스트 항목을 추가하여 리전 간 복제를 확인합니다.
- 글로벌 테이블 구성 및 복제본 상태를 설명합니다.

SDK for Java 2.x

를 사용하여 글로벌 보조 인덱스 및 DynamoDB 스트림으로 테이블을 생성합니다 AWS SDK for Java 2.x.

```
    public static CreateTableResponse createTableWithGSI(
        final DynamoDbClient dynamoDbClient, final String tableName, final String
indexName) {

        if (dynamoDbClient == null) {
```

```
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (indexName == null || indexName.trim().isEmpty()) {
            throw new IllegalArgumentException("Index name cannot be null or
empty");
        }

        try {
            LOGGER.info("Creating table: " + tableName + " with GSI: " + indexName);

            CreateTableRequest createTableRequest = CreateTableRequest.builder()
                .tableName(tableName)
                .attributeDefinitions(
                    AttributeDefinition.builder()
                        .attributeName("Artist")
                        .attributeType(ScalarAttributeType.S)
                        .build(),
                    AttributeDefinition.builder()
                        .attributeName("SongTitle")
                        .attributeType(ScalarAttributeType.S)
                        .build())
                .keySchema(
                    KeySchemaElement.builder()
                        .attributeName("Artist")
                        .keyType(KeyType.HASH)
                        .build(),
                    KeySchemaElement.builder()
                        .attributeName("SongTitle")
                        .keyType(KeyType.RANGE)
                        .build())
                .billingMode(BillingMode.PAY_PER_REQUEST)
                .globalSecondaryIndexes(GlobalSecondaryIndex.builder()
                    .indexName(indexName)
                    .keySchema(KeySchemaElement.builder()
                        .attributeName("SongTitle")
                        .keyType(KeyType.HASH)
                        .build())
                    .projection(

Projection.builder().projectionType(ProjectionType.ALL).build())
```

```
                    .build())
                .streamSpecification(StreamSpecification.builder()
                    .streamEnabled(true)
                    .streamViewType(StreamViewType.NEW_AND_OLD_IMAGES)
                    .build())
                .build();

            CreateTableResponse response =
dynamoDbClient.createTable(createTableRequest);
            LOGGER.info("Table creation initiated. Status: "
                + response.tableDescription().tableStatus());

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to create table: " + tableName + " - " +
e.getMessage());
            throw e;
        }
    }
```

를 사용하여 테이블이 활성화될 때까지 기다립니다 AWS SDK for Java 2.x.

```
    public static void waitForTableActive(final DynamoDbClient dynamoDbClient, final
String tableName) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }

        try {
            LOGGER.info("Waiting for table to become active: " + tableName);

            try (DynamoDbWaiter waiter =
                DynamoDbWaiter.builder().client(dynamoDbClient).build()) {
                DescribeTableRequest request =
                    DescribeTableRequest.builder().tableName(tableName).build();
```

```
                waiter.waitUntilTableExists(request);
                LOGGER.info("Table is now active: " + tableName);
            }

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to wait for table to become active: " + tableName
+ " - " + e.getMessage());
            throw e;
        }
    }
```

를 사용하여 글로벌 테이블을 생성하거나 확장할 복제본을 추가합니다 AWS SDK for Java 2.x.

```
    public static UpdateTableResponse addReplica(
        final DynamoDbClient dynamoDbClient,
        final String tableName,
        final Region replicaRegion,
        final String indexName,
        final Long readCapacity) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (replicaRegion == null) {
            throw new IllegalArgumentException("Replica region cannot be null");
        }
        if (indexName == null || indexName.trim().isEmpty()) {
            throw new IllegalArgumentException("Index name cannot be null or
empty");
        }
        if (readCapacity == null || readCapacity <= 0) {
            throw new IllegalArgumentException("Read capacity must be a positive
number");
        }

        try {
            LOGGER.info("Adding replica in region: " + replicaRegion.id() + " for
table: " + tableName);
```

```
            // Create a ReplicationGroupUpdate for adding a replica
            ReplicationGroupUpdate replicationGroupUpdate =
ReplicationGroupUpdate.builder()
                .create(builder -> builder.regionName(replicaRegion.id())
                    .globalSecondaryIndexes(ReplicaGlobalSecondaryIndex.builder()
                        .indexName(indexName)

.provisionedThroughputOverride(ProvisionedThroughputOverride.builder()
                            .readCapacityUnits(readCapacity)
                            .build())
                        .build())
                    .build())
                .build();

            UpdateTableRequest updateTableRequest = UpdateTableRequest.builder()
                .tableName(tableName)
                .replicaUpdates(replicationGroupUpdate)
                .build();

            UpdateTableResponse response =
dynamoDbClient.updateTable(updateTableRequest);
            LOGGER.info("Replica addition initiated in region: " +
replicaRegion.id());

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to add replica in region: " + replicaRegion.id() +
" - " + e.getMessage());
            throw e;
        }
    }
```

를 사용하여 글로벌 테이블에서 복제본을 제거합니다 AWS SDK for Java 2.x.

```
    public static UpdateTableResponse removeReplica(
        final DynamoDbClient dynamoDbClient, final String tableName, final Region
replicaRegion) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
```

```
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (replicaRegion == null) {
            throw new IllegalArgumentException("Replica region cannot be null");
        }

        try {
            LOGGER.info("Removing replica in region: " + replicaRegion.id() + " for
table: " + tableName);

            // Create a ReplicationGroupUpdate for removing a replica
            ReplicationGroupUpdate replicationGroupUpdate =
ReplicationGroupUpdate.builder()
                .delete(builder -> builder.regionName(replicaRegion.id()).build())
                .build();

            UpdateTableRequest updateTableRequest = UpdateTableRequest.builder()
                .tableName(tableName)
                .replicaUpdates(replicationGroupUpdate)
                .build();

            UpdateTableResponse response =
dynamoDbClient.updateTable(updateTableRequest);
            LOGGER.info("Replica removal initiated in region: " +
replicaRegion.id());

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to remove replica in region: " +
replicaRegion.id() + " - " + e.getMessage());
            throw e;
        }
    }
```

를 사용하여 복제를 확인하려면 테스트 항목을 추가합니다 AWS SDK for Java 2.x.

```
    public static PutItemResponse putTestItem(
```

```
        final DynamoDbClient dynamoDbClient, final String tableName, final String
artist, final String songTitle) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (artist == null || artist.trim().isEmpty()) {
            throw new IllegalArgumentException("Artist cannot be null or empty");
        }
        if (songTitle == null || songTitle.trim().isEmpty()) {
            throw new IllegalArgumentException("Song title cannot be null or
empty");
        }

        try {
            LOGGER.info("Adding test item to table: " + tableName);

            Map<String,
software.amazon.awssdk.services.dynamodb.model.AttributeValue> item = new
HashMap<>();
            item.put(
                "Artist",

software.amazon.awssdk.services.dynamodb.model.AttributeValue.builder()
                    .s(artist)
                    .build());
            item.put(
                "SongTitle",

software.amazon.awssdk.services.dynamodb.model.AttributeValue.builder()
                    .s(songTitle)
                    .build());

            PutItemRequest putItemRequest =
                PutItemRequest.builder().tableName(tableName).item(item).build();

            PutItemResponse response = dynamoDbClient.putItem(putItemRequest);
            LOGGER.info("Test item added successfully");

            return response;
```

```
        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to add test item to table: " + tableName + " - " +
e.getMessage());
            throw e;
        }
    }
```

를 사용하여 전역 테이블 구성 및 복제본을 설명합니다 AWS SDK for Java 2.x.

```
    public static DescribeTableResponse describeTable(final DynamoDbClient
dynamoDbClient, final String tableName) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }

        try {
            LOGGER.info("Describing table: " + tableName);

            DescribeTableRequest request =
                DescribeTableRequest.builder().tableName(tableName).build();

            DescribeTableResponse response = dynamoDbClient.describeTable(request);

            LOGGER.info("Table status: " + response.table().tableStatus());
            if (response.table().replicas() != null
                && !response.table().replicas().isEmpty()) {
                LOGGER.info("Number of replicas: " +
response.table().replicas().size());
                response.table()
                    .replicas()
                    .forEach(replica -> LOGGER.info(
                        "Replica region: " + replica.regionName() + ", Status: " +
replica.replicaStatus()));
            }

            return response;
```

```
        } catch (ResourceNotFoundException e) {
            LOGGER.severe("Table not found: " + tableName + " - " + e.getMessage());
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to describe table: " + tableName + " - " +
e.getMessage());
            throw e;
        }
    }
```

를 사용한 글로벌 테이블 작업의 전체 예제입니다 AWS SDK for Java 2.x.

```
    public static void exampleUsage(final Region sourceRegion, final Region
replicaRegion) {

        String tableName = "Music";
        String indexName = "SongTitleIndex";
        Long readCapacity = 15L;

        // Create DynamoDB client for the source region
        try (DynamoDbClient dynamoDbClient =
            DynamoDbClient.builder().region(sourceRegion).build()) {

            try {
                // Step 1: Create the initial table with GSI and streams
                LOGGER.info("Step 1: Creating table in source region: " +
sourceRegion.id());
                createTableWithGSI(dynamoDbClient, tableName, indexName);

                // Step 2: Wait for table to become active
                LOGGER.info("Step 2: Waiting for table to become active");
                waitForTableActive(dynamoDbClient, tableName);

                // Step 3: Add replica in destination region
                LOGGER.info("Step 3: Adding replica in region: " +
replicaRegion.id());
                addReplica(dynamoDbClient, tableName, replicaRegion, indexName,
readCapacity);

                // Step 4: Wait a moment for replica creation to start
                Thread.sleep(5000);
```

```
                // Step 5: Describe table to view replica information
                LOGGER.info("Step 5: Describing table to view replicas");
                describeTable(dynamoDbClient, tableName);

                // Step 6: Add a test item to verify replication
                LOGGER.info("Step 6: Adding test item to verify replication");
                putTestItem(dynamoDbClient, tableName, "TestArtist", "TestSong");

                LOGGER.info("Global table setup completed successfully!");
                LOGGER.info("You can verify replication by checking the item in
region: " + replicaRegion.id());

                // Step 7: Remove replica and clean up table
                LOGGER.info("Step 7: Removing replica from region: " +
replicaRegion.id());
                removeReplica(dynamoDbClient, tableName, replicaRegion);
                DeleteTableResponse deleteTableResponse =
dynamoDbClient.deleteTable(
                    DeleteTableRequest.builder().tableName(tableName).build());
                LOGGER.info("MREC global table demonstration completed
successfully!");

            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                throw new RuntimeException("Thread was interrupted", e);
            } catch (DynamoDbException e) {
                LOGGER.severe("DynamoDB operation failed: " + e.getMessage());
                throw e;
            }
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - CreateTable
  - DescribeTable
  - PutItem
  - UpdateTable

이미지에서 PPE 감지

다음 코드 예제에서는 Amazon Rekognition을 사용하여 이미지에서 개인 보호 장비(PPE)를 감지하는 앱을 구축하는 방법을 보여줍니다.

SDK for Java 2.x

개인 보호 장비로 이미지를 감지하는 AWS Lambda 함수를 생성하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

DynamoDB 성능 모니터링

다음 코드 예제는 성능 모니터링을 위해 애플리케이션의 DynamoDB 사용을 구성하는 방법을 보여줍니다.

SDK for Java 2.x

이 예제는 DynamoDB의 성능을 모니터링하도록 Java 애플리케이션을 구성하는 방법을 보여줍니다. 애플리케이션은 성능을 모니터링할 수 있는 CloudWatch로 지표 데이터를 전송합니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- CloudWatch
- DynamoDB

고급 쿼리 작업 수행

다음 코드 예제에서는 DynamoDB에서 고급 쿼리 작업을 수행하는 방법을 보여줍니다.

- 다양한 필터링 및 조건 기법을 사용하여 테이블을 쿼리합니다.

- 대규모 결과 세트에 페이지 매김을 구현합니다.
- 대체 액세스 패턴에 글로벌 보조 인덱스를 사용합니다.
- 애플리케이션 요구 사항에 따라 일관성 제어를 적용합니다.

SDK for Java 2.x

를 사용하여 강력히 일관된 읽기로 쿼리합니다 AWS SDK for Java 2.x.

```
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 java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithConsistentReads(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final boolean useConsistentRead) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
```

```
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .consistentRead(useConsistentRead)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query successful. Found {0} items",
 response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with consistent reads", e);
            throw e;
        }
    }
```

와 함께 글로벌 보조 인덱스를 사용하여 쿼리합니다 AWS SDK for Java 2.x.

```
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 java.util.HashMap;
import java.util.Map;

    public QueryResponse queryTable(
        final String tableName, final String partitionKeyName, final String
 partitionKeyValue) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
```

```
        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query on base table successful. Found " +
response.count() + " items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw new DynamoDbQueryException("Table not found: " + tableName, e);
        } catch (DynamoDbException e) {
            System.err.println("Error querying base table: " + e.getMessage());
            throw new DynamoDbQueryException("Failed to execute query on base
table", e);
        }
    }

    /**
     * Queries a DynamoDB Global Secondary Index (GSI) by partition key.
     *
     * @param tableName         The name of the DynamoDB table
     * @param indexName         The name of the GSI
     * @param partitionKeyName  The name of the GSI partition key attribute
     * @param partitionKeyValue The value of the GSI partition key to query
     * @return The query response from DynamoDB
```

```
     * @throws ResourceNotFoundException if the table or index doesn't exist
     * @throws DynamoDbException if the query fails
     */
    public QueryResponse queryGlobalSecondaryIndex(
        final String tableName, final String indexName, final String
partitionKeyName, final String partitionKeyValue) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Index name", indexName);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_IK,
partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_IK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .indexName(indexName)
            .keyConditionExpression(GSI_KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query on GSI successful. Found " + response.count()
+ " items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format(
                "Error: The Amazon DynamoDB table \"%s\" or index \"%s\" can't be
found.\n", tableName, indexName);
            throw new DynamoDbQueryException("Table or index not found: " +
tableName + "/" + indexName, e);
        } catch (DynamoDbException e) {
```

```
            System.err.println("Error querying GSI: " + e.getMessage());
            throw new DynamoDbQueryException("Failed to execute query on GSI", e);
        }
    }
```

를 사용하여 페이지 매김으로 쿼리합니다 AWS SDK for Java 2.x.

```
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 java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    public List<Map<String, AttributeValue>> queryWithPagination(
        final String tableName, final String partitionKeyName, final String
 partitionKeyValue, final int pageSize) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validatePositiveInteger("Page size", pageSize);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        QueryRequest.Builder queryRequestBuilder = QueryRequest.builder()
```

```
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .limit(pageSize);

        // List to store all items from all pages
        final List<Map<String, AttributeValue>> allItems = new ArrayList<>();

        // Map to store the last evaluated key for pagination
        Map<String, AttributeValue> lastEvaluatedKey = null;
        int pageNumber = 1;

        try {
            do {
                // If we have a last evaluated key, use it for the next page
                if (lastEvaluatedKey != null) {
                    queryRequestBuilder.exclusiveStartKey(lastEvaluatedKey);
                }

                // Execute the query
                final QueryResponse response =
dynamoDbClient.query(queryRequestBuilder.build());

                // Process the current page of results
                final List<Map<String, AttributeValue>> pageItems =
response.items();
                allItems.addAll(pageItems);

                // Get the last evaluated key for the next page
                lastEvaluatedKey = response.lastEvaluatedKey();
                if (lastEvaluatedKey != null && lastEvaluatedKey.isEmpty()) {
                    lastEvaluatedKey = null;
                }

                System.out.println("Page " + pageNumber + ": Retrieved " +
pageItems.size() + " items (Running total: "
                    + allItems.size() + ")");

                pageNumber++;

            } while (lastEvaluatedKey != null);
```

```
            System.out.println("Query with pagination complete. Retrieved a total of
 " + allItems.size()
                + " items across " + (pageNumber - 1) + " pages");

            return allItems;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
 found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with pagination: " + e.getMessage());
            throw e;
        }
    }
```

를 사용하여 복잡한 필터로 쿼리합니다 AWS SDK for Java 2.x.

```
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 java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithComplexFilter(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String statusAttrName,
        final String activeStatus,
        final String pendingStatus,
        final String priceAttrName,
        final double minPrice,
        final double maxPrice,
        final String categoryAttrName) {
```

```
        // Validate parameters
        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Status attribute name",
statusAttrName);
        CodeSampleUtils.validateStringParameter("Active status", activeStatus);
        CodeSampleUtils.validateStringParameter("Pending status", pendingStatus);
        CodeSampleUtils.validateStringParameter("Price attribute name",
priceAttrName);
        CodeSampleUtils.validateStringParameter("Category attribute name",
categoryAttrName);
        CodeSampleUtils.validateNumericRange("Minimum price", minPrice, 0.0,
Double.MAX_VALUE);
        CodeSampleUtils.validateNumericRange("Maximum price", maxPrice, minPrice,
Double.MAX_VALUE);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#pk", partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_STATUS,
statusAttrName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PRICE,
priceAttrName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_CATEGORY,
categoryAttrName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            ":pkValue", AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_ACTIVE,
            AttributeValue.builder().s(activeStatus).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PENDING,
            AttributeValue.builder().s(pendingStatus).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_MIN_PRICE,
            AttributeValue.builder().n(String.valueOf(minPrice)).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_MAX_PRICE,
            AttributeValue.builder().n(String.valueOf(maxPrice)).build());
```

```
        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(FILTER_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        return dynamoDbClient.query(queryRequest);
    }
```

를 사용하여 동적으로 구성된 필터 표현식을 사용하여 쿼리합니다 AWS SDK for Java 2.x.

```
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 java.util.HashMap;
import java.util.Map;

    public static QueryResponse queryWithDynamicFilter(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final Map<String, Object> filterCriteria,
        final Region region,
        final DynamoDbClient dynamoDbClient) {

        validateParameters(tableName, partitionKeyName, partitionKeyValue,
 filterCriteria);

        DynamoDbClient ddbClient = dynamoDbClient;
        boolean shouldClose = false;

        try {
            if (ddbClient == null) {
                ddbClient = createClient(region);
```

```
                shouldClose = true;
            }

            final QueryWithDynamicFilter queryHelper = new
QueryWithDynamicFilter(ddbClient);
            return queryHelper.queryWithDynamicFilter(tableName, partitionKeyName,
partitionKeyValue, filterCriteria);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Failed to execute dynamic filter query: " +
e.getMessage());
            throw e;
        } catch (Exception e) {
            System.err.println("Unexpected error during query: " + e.getMessage());
            throw e;
        } finally {
            if (shouldClose && ddbClient != null) {
                ddbClient.close();
            }
        }
    }

    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue>
<filterAttrName> <filterAttrValue> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    filterAttrName - The name of the attribute to filter on.
                    filterAttrValue - The value to filter by.
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 5) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        final String tableName = args[0];
        final String partitionKeyName = args[1];
        final String partitionKeyValue = args[2];
        final String filterAttrName = args[3];
        final String filterAttrValue = args[4];
        final Region region = args.length > 5 ? Region.of(args[5]) :
Region.US_EAST_1;

        System.out.println("Querying items with dynamic filter: " + filterAttrName +
" = " + filterAttrValue);

        try {
            // Using the builder pattern to create and execute the query
            final QueryResponse response = new DynamicFilterQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withFilterCriterion(filterAttrName, filterAttrValue)
                .withRegion(region)
                .execute();

            // Process the results
            System.out.println("Found " + response.count() + " items:");
            response.items().forEach(item -> System.out.println(item));

            // Demonstrate multiple filter criteria
            System.out.println("\nNow querying with multiple filter criteria:");

            Map<String, Object> multipleFilters = new HashMap<>();
            multipleFilters.put(filterAttrName, filterAttrValue);
            multipleFilters.put("status", "active");

            final QueryResponse multiFilterResponse = new
DynamicFilterQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withFilterCriteria(multipleFilters)
                .withRegion(region)
                .execute();

            System.out.println("Found " + multiFilterResponse.count() + " items with
multiple filters:");
```

```
            multiFilterResponse.items().forEach(item -> System.out.println(item));

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

를 사용하여 필터 표현식 및 제한으로 쿼리합니다 AWS SDK for Java 2.x.

```
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 java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithFilterAndLimit(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String filterAttrName,
        final String filterAttrValue,
        final int limit) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
```

```
        CodeSampleUtils.validateStringParameter("Filter attribute name",
filterAttrName);
        CodeSampleUtils.validateStringParameter("Filter attribute value",
filterAttrValue);
        CodeSampleUtils.validatePositiveInteger("Limit", limit);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_FILTER,
filterAttrName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_FILTER,
            AttributeValue.builder().s(filterAttrValue).build());

        // Create the filter expression
        final String filterExpression = "#filterAttr = :filterValue";

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(filterExpression)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .limit(limit)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query with filter and limit successful. Found
{0} items", response.count());
            LOGGER.log(
                Level.INFO, "ScannedCount: {0} (total items evaluated before
filtering)", response.scannedCount());
            return response;
```

```
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with filter and limit: {0}",
e.getMessage());
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Query를 참조하세요.

목록 작업 수행

다음 코드 예제에서는 DynamoDB에서 목록 작업을 수행하는 방법을 보여줍니다.

- 목록 속성에 요소를 추가합니다.
- 목록 속성에서 요소를 제거합니다.
- 인덱스별로 목록의 특정 요소를 업데이트합니다.
- 목록 추가 및 목록 인덱스 함수를 사용합니다.

SDK for Java 2.x

를 사용하여 목록 작업을 시연합니다 AWS SDK for Java 2.x.

```
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.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    /**
     * Appends items to a list attribute.
```

```
     *
     * <p>This method demonstrates how to use the list_append function to add
     * items to the end of a list attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param listAttributeName The name of the list attribute
     * @param itemsToAppend The items to append to the list
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse appendToList(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String listAttributeName,
        List<AttributeValue> itemsToAppend) {

        // Create a list value from the items to append
        AttributeValue listValue =
AttributeValue.builder().l(itemsToAppend).build();

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attrName =
list_append(if_not_exists(#attrName, :emptyList), :newItems)")
            .expressionAttributeNames(Map.of("#attrName", listAttributeName))
            .expressionAttributeValues(Map.of(
                ":newItems",
                listValue,
                ":emptyList",
                AttributeValue.builder().l(new
ArrayList<AttributeValue>()).build()))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
```

```
     * Prepends items to a list attribute.
     *
     * <p>This method demonstrates how to use the list_append function to add
     * items to the beginning of a list attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param listAttributeName The name of the list attribute
     * @param itemsToPrepend The items to prepend to the list
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse prependToList(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String listAttributeName,
        List<AttributeValue> itemsToPrepend) {

        // Create a list value from the items to prepend
        AttributeValue listValue =
AttributeValue.builder().l(itemsToPrepend).build();

        // Define the update parameters
        // Note: To prepend, we put the new items first in the list_append function
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attrName = list_append(:newItems,
if_not_exists(#attrName, :emptyList))")
            .expressionAttributeNames(Map.of("#attrName", listAttributeName))
            .expressionAttributeValues(Map.of(
                ":newItems",
                listValue,
                ":emptyList",
                AttributeValue.builder().l(new
ArrayList<AttributeValue>()).build()))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }
```

```
/**
 * Updates a specific element in a list attribute.
 *
 * <p>This method demonstrates how to update a specific element in a list
 * by its index.
 *
 * @param dynamoDbClient The DynamoDB client
 * @param tableName The name of the DynamoDB table
 * @param key The key of the item to update
 * @param listAttributeName The name of the list attribute
 * @param index The index of the element to update
 * @param newValue The new value for the element
 * @return The response from DynamoDB
 * @throws DynamoDbException if an error occurs during the operation
 */
public static UpdateItemResponse updateListElement(
    DynamoDbClient dynamoDbClient,
    String tableName,
    Map<String, AttributeValue> key,
    String listAttributeName,
    int index,
    AttributeValue newValue) {

    // Define the update parameters
    UpdateItemRequest request = UpdateItemRequest.builder()
        .tableName(tableName)
        .key(key)
        .updateExpression("SET #attrName[" + index + "] = :newValue")
        .expressionAttributeNames(Map.of("#attrName", listAttributeName))
        .expressionAttributeValues(Map.of(":newValue", newValue))
        .returnValues("UPDATED_NEW")
        .build();

    // Perform the update operation
    return dynamoDbClient.updateItem(request);
}

/**
 * Removes a specific element from a list attribute.
 *
 * <p>This method demonstrates how to remove a specific element from a list
 * by its index.
 *
```

```
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param listAttributeName The name of the list attribute
     * @param index The index of the element to remove
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse removeListElement(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String listAttributeName,
        int index) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("REMOVE #attrName[" + index + "]")
            .expressionAttributeNames(Map.of("#attrName", listAttributeName))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Gets the current value of a list attribute.
     *
     * <p>Helper method to retrieve the current value of a list attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @param listAttributeName The name of the list attribute
     * @return The list attribute value or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static List<AttributeValue> getListAttribute(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, String listAttributeName) {
```

```
        // Define the get parameters
        GetItemRequest request = GetItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .projectionExpression(listAttributeName)
            .build();

        try {
            // Perform the get operation
            GetItemResponse response = dynamoDbClient.getItem(request);

            // Return the list attribute if it exists, otherwise null
            if (response.item() != null &&
response.item().containsKey(listAttributeName)) {
                return response.item().get(listAttributeName).l();
            }

            return null;
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to get list attribute: " + e.getMessage())
                .cause(e)
                .build();
        }
    }
```

를 사용한 목록 작업 사용의 예입니다 AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating list operations in DynamoDB");

        try {
            // Example 1: Append items to a list
            System.out.println("\nExample 1: Appending items to a list");
            List<AttributeValue> tagsToAppend = List.of(
                AttributeValue.builder().s("Electronics").build(),
                AttributeValue.builder().s("Gadget").build());
```

```
            UpdateItemResponse appendResponse = appendToList(dynamoDbClient,
tableName, key, "Tags", tagsToAppend);

            System.out.println("Updated list attribute: " +
appendResponse.attributes());

            // Example 2: Prepend items to a list
            System.out.println("\nExample 2: Prepending items to a list");
            List<AttributeValue> tagsToPrepend = List.of(
                AttributeValue.builder().s("Featured").build(),
                AttributeValue.builder().s("New").build());

            UpdateItemResponse prependResponse = prependToList(dynamoDbClient,
tableName, key, "Tags", tagsToPrepend);

            System.out.println("Updated list attribute: " +
prependResponse.attributes());

            // Example 3: Update a specific element in a list
            System.out.println("\nExample 3: Updating a specific element in a
list");
            UpdateItemResponse updateResponse = updateListElement(
                dynamoDbClient,
                tableName,
                key,
                "Tags",
                0,
                AttributeValue.builder().s("BestSeller").build());

            System.out.println("Updated list attribute: " +
updateResponse.attributes());

            // Example 4: Remove a specific element from a list
            System.out.println("\nExample 4: Removing a specific element from a
list");
            UpdateItemResponse removeResponse = removeListElement(dynamoDbClient,
tableName, key, "Tags", 1);

            System.out.println("Updated list attribute: " +
removeResponse.attributes());

            // Example 5: Get the current value of a list attribute
```

```
            System.out.println("\nExample 5: Getting the current value of a list
attribute");
            List<AttributeValue> currentList = getListAttribute(dynamoDbClient,
tableName, key, "Tags");

            if (currentList != null) {
                System.out.println("Current list attribute:");
                for (int i = 0; i < currentList.size(); i++) {
                    System.out.println("  [" + i + "]: " + currentList.get(i).s());
                }
            } else {
                System.out.println("List attribute not found");
            }

            // Explain list operations
            System.out.println("\nKey points about DynamoDB list operations:");
            System.out.println("1. Lists are ordered collections of attributes");
            System.out.println("2. Use list_append to add items to a list");
            System.out.println("3. To append items, use list_append(existingList,
newItems)");
            System.out.println("4. To prepend items, use list_append(newItems,
existingList)");
            System.out.println("5. Use index notation (list[0]) to access or update
specific elements");
            System.out.println("6. Use REMOVE to delete elements from a list");
            System.out.println("7. List indices are zero-based");
            System.out.println("8. Use if_not_exists to handle the case where the
list doesn't exist yet");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateItem을 참조하세요.

맵 작업 수행

다음 코드 예제에서는 DynamoDB에서 맵 작업을 수행하는 방법을 보여줍니다.

- 맵 구조에서 중첩 속성을 추가하고 업데이트합니다.

- 맵에서 특정 필드를 제거합니다.
- 깊이 중첩된 맵 속성을 사용하여 작업합니다.

SDK for Java 2.x

를 사용하여 맵 작업을 시연합니다 AWS SDK for Java 2.x.

```
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.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

    /**
     * Updates a map attribute that may not exist.
     *
     * <p>This method demonstrates how to safely update a map attribute
     * by using if_not_exists to handle the case where the map doesn't exist yet.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param mapName The name of the map attribute
     * @param mapKey The key within the map to update
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateMapAttributeSafe(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String mapName,
        String mapKey,
        AttributeValue value) {
```

```
        // Create an empty map to use if the map doesn't exist
        Map<String, AttributeValue> emptyMap = new HashMap<>();
        AttributeValue emptyMapValue = AttributeValue.builder().m(emptyMap).build();

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #mapName = if_not_exists(#mapName, :emptyMap),
#mapName.#mapKey = :value")
            .expressionAttributeNames(Map.of(
                "#mapName", mapName,
                "#mapKey", mapKey))
            .expressionAttributeValues(Map.of(
                ":value",
                value,
                ":emptyMap",
                AttributeValue.builder().m(new HashMap<>()).build()))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Adds an attribute to a nested map.
     *
     * <p>This method demonstrates how to update a nested attribute without
     * overwriting the entire map.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param path The path to the nested attribute as a list
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse addToNestedMap(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        List<String> path,
```

```
        AttributeValue value) {

        // Create expression attribute names for each part of the path
        Map<String, String> expressionAttributeNames = new HashMap<>();
        for (int i = 0; i < path.size(); i++) {
            expressionAttributeNames.put("#attr" + i, path.get(i));
        }

        // Build the attribute path using the expression attribute names
        StringBuilder attributePathExpression = new StringBuilder();
        for (int i = 0; i < path.size(); i++) {
            if (i > 0) {
                attributePathExpression.append(".");
            }
            attributePathExpression.append("#attr").append(i);
        }

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET " + attributePathExpression.toString() + "
= :value")
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Removes an attribute from a map.
     *
     * <p>This method demonstrates how to remove a specific attribute from a map.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param mapName The name of the map attribute
     * @param mapKey The key within the map to remove
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
```

```
     */
    public static UpdateItemResponse removeMapAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String mapName,
        String mapKey) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("REMOVE #mapName.#mapKey")
            .expressionAttributeNames(Map.of(
                "#mapName", mapName,
                "#mapKey", mapKey))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Creates a map with multiple attributes in a single operation.
     *
     * <p>This method demonstrates how to create a map with multiple attributes
     * in a single update operation.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param mapName The name of the map attribute
     * @param attributes The attributes to set in the map
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse createMapWithAttributes(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String mapName,
        Map<String, AttributeValue> attributes) {
```

```
        // Create a map value from the attributes
        AttributeValue mapValue = AttributeValue.builder().m(attributes).build();

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #mapName = :mapValue")
            .expressionAttributeNames(Map.of("#mapName", mapName))
            .expressionAttributeValues(Map.of(":mapValue", mapValue))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Gets the current value of a map attribute.
     *
     * <p>Helper method to retrieve the current value of a map attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @param mapName The name of the map attribute
     * @return The map attribute value or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Map<String, AttributeValue> getMapAttribute(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, String mapName) {

        // Define the get parameters
        GetItemRequest request = GetItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .projectionExpression(mapName)
            .build();

        try {
            // Perform the get operation
            GetItemResponse response = dynamoDbClient.getItem(request);
```

```
            // Return the map attribute if it exists, otherwise null
            if (response.item() != null && response.item().containsKey(mapName)) {
                return response.item().get(mapName).m();
            }

            return null;
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to get map attribute: " + e.getMessage())
                .cause(e)
                .build();
        }
    }
```

를 사용한 맵 작업 사용의 예입니다 AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating map operations in DynamoDB");

        try {
            // Example 1: Create a map with multiple attributes
            System.out.println("\nExample 1: Creating a map with multiple
attributes");
            Map<String, AttributeValue> productDetails = new HashMap<>();
            productDetails.put("Color", AttributeValue.builder().s("Red").build());
            productDetails.put("Weight", AttributeValue.builder().n("2.5").build());
            productDetails.put(
                "Dimensions", AttributeValue.builder().s("10x20x5").build());

            UpdateItemResponse createResponse =
                createMapWithAttributes(dynamoDbClient, tableName, key, "Details",
productDetails);

            System.out.println("Created map attribute: " +
createResponse.attributes());

            // Example 2: Update a specific attribute in a map
```

```
            System.out.println("\nExample 2: Updating a specific attribute in a
map");
            UpdateItemResponse updateResponse = updateMapAttributeSafe(
                dynamoDbClient,
                tableName,
                key,
                "Details",
                "Color",
                AttributeValue.builder().s("Blue").build());

            System.out.println("Updated map attribute: " +
updateResponse.attributes());

            // Example 3: Add an attribute to a nested map
            System.out.println("\nExample 3: Adding an attribute to a nested map");
            UpdateItemResponse nestedResponse = addToNestedMap(
                dynamoDbClient,
                tableName,
                key,
                List.of("Specifications", "Technical", "Resolution"),
                AttributeValue.builder().s("1920x1080").build());

            System.out.println("Added to nested map: " +
nestedResponse.attributes());

            // Example 4: Remove an attribute from a map
            System.out.println("\nExample 4: Removing an attribute from a map");
            UpdateItemResponse removeResponse =
                removeMapAttribute(dynamoDbClient, tableName, key, "Details",
"Dimensions");

            System.out.println("Updated map after removal: " +
removeResponse.attributes());

            // Example 5: Get the current value of a map attribute
            System.out.println("\nExample 5: Getting the current value of a map
attribute");
            Map<String, AttributeValue> currentMap = getMapAttribute(dynamoDbClient,
tableName, key, "Details");

            if (currentMap != null) {
                System.out.println("Current map attribute:");
                for (Map.Entry<String, AttributeValue> entry :
currentMap.entrySet()) {
```

```
                    System.out.println("  " + entry.getKey() + ": " +
entry.getValue());
                }
            } else {
                System.out.println("Map attribute not found");
            }

            // Explain map operations
            System.out.println("\nKey points about DynamoDB map operations:");
            System.out.println("1. Maps are unordered collections of name-value
pairs");
            System.out.println("2. Use dot notation (map.key) to access or update
specific attributes");
            System.out.println("3. You can update individual attributes without
overwriting the entire map");
            System.out.println("4. Maps can be nested to create complex data
structures");
            System.out.println("5. Use REMOVE to delete attributes from a map");
            System.out.println("6. You can create a map with multiple attributes in
a single operation");
            System.out.println("7. Map keys are case-sensitive");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateItem을 참조하세요.

집합 작업 수행

다음 코드 예제에서는 DynamoDB에서 집합 작업을 수행하는 방법을 보여줍니다.

- 집합 속성에 요소를 추가합니다.
- 집합 속성에서 요소를 제거합니다.
- 집합에 ADD 및 DELETE 작업을 사용합니다.

SDK for Java 2.x

를 사용하여 세트 작업을 시연합니다 AWS SDK for Java 2.x.

```
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.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ReturnValue;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;

    /**
     * Adds values to a string set attribute.
     *
     * <p>This method demonstrates how to use the ADD operation to add values
     * to a string set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param setAttributeName The name of the set attribute
     * @param valuesToAdd The values to add to the set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse addToStringSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        Set<String> valuesToAdd) {

        // Create a string set value from the values to add
        AttributeValue setValue = AttributeValue.builder().ss(valuesToAdd).build();

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
```

```
            .tableName(tableName)
            .key(key)
            .updateExpression("ADD #setAttr :valuesToAdd")
            .expressionAttributeNames(Map.of("#setAttr", setAttributeName))
            .expressionAttributeValues(Map.of(":valuesToAdd", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Adds values to a number set attribute.
     *
     * <p>This method demonstrates how to use the ADD operation to add values
     * to a number set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param setAttributeName The name of the set attribute
     * @param valuesToAdd The values to add to the set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse addToNumberSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        Set<Number> valuesToAdd) {

        // Convert numbers to strings for DynamoDB
        Set<String> stringValues = new HashSet<>();
        for (Number value : valuesToAdd) {
            stringValues.add(value.toString());
        }

        // Create a number set value from the values to add
        AttributeValue setValue = AttributeValue.builder().ns(stringValues).build();

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
```

```
            .tableName(tableName)
            .key(key)
            .updateExpression("ADD #setAttr :valuesToAdd")
            .expressionAttributeNames(Map.of("#setAttr", setAttributeName))
            .expressionAttributeValues(Map.of(":valuesToAdd", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Removes values from a set attribute.
     *
     * <p>This method demonstrates how to use the DELETE operation to remove values
     * from a set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param setAttributeName The name of the set attribute
     * @param valuesToRemove The values to remove from the set
     * @param isNumberSet Whether the set is a number set (true) or string set
(false)
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse removeFromSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        Set<?> valuesToRemove,
        boolean isNumberSet) {

        AttributeValue setValue;

        if (isNumberSet) {
            // Convert numbers to strings for DynamoDB
            Set<String> stringValues = new HashSet<>();
            for (Object value : valuesToRemove) {
                if (value instanceof Number) {
                    stringValues.add(value.toString());
```

```
                } else {
                    throw new IllegalArgumentException("Values must be numbers for a
number set");
                }
            }

            setValue = AttributeValue.builder().ns(stringValues).build();
        } else {
            // Convert objects to strings for DynamoDB
            Set<String> stringValues = new HashSet<>();
            for (Object value : valuesToRemove) {
                stringValues.add(value.toString());
            }

            setValue = AttributeValue.builder().ss(stringValues).build();
        }

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("DELETE #setAttr :valuesToRemove")
            .expressionAttributeNames(Map.of("#setAttr", setAttributeName))
            .expressionAttributeValues(Map.of(":valuesToRemove", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Checks if a value exists in a set attribute.
     *
     * <p>This method demonstrates how to use the contains function to check
     * if a value exists in a set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to check
     * @param setAttributeName The name of the set attribute
     * @param valueToCheck The value to check for
     * @return Map containing the result of the check
     * @throws DynamoDbException if an error occurs during the operation
```

```
     */
    public static Map<String, Object> checkIfValueInSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        String valueToCheck) {

        Map<String, Object> result = new HashMap<>();

        try {
            // Define the update parameters with a condition expression
            UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #tempAttr = :tempVal")
                .conditionExpression("contains(#setAttr, :valueToCheck)")
                .expressionAttributeNames(Map.of("#setAttr", setAttributeName,
"#tempAttr", "TempAttribute"))
                .expressionAttributeValues(Map.of(
                    ":valueToCheck",
AttributeValue.builder().s(valueToCheck).build(),
                    ":tempVal", AttributeValue.builder().s("TempValue").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            // Attempt the update operation
            dynamoDbClient.updateItem(request);

            // If we get here, the condition was met
            result.put("exists", true);
            result.put("message", "Value '" + valueToCheck + "' exists in the set");

            // Clean up the temporary attribute
            UpdateItemRequest cleanupRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("REMOVE #tempAttr")
                .expressionAttributeNames(Map.of("#tempAttr", "TempAttribute"))
                .build();

            dynamoDbClient.updateItem(cleanupRequest);

        } catch (DynamoDbException e) {
```

```
            if (e.getMessage().contains("ConditionalCheckFailed")) {
                // The condition was not met
                result.put("exists", false);
                result.put("message", "Value '" + valueToCheck + "' does not exist
in the set");
            } else {
                // Some other error occurred
                result.put("exists", false);
                result.put("message", "Error checking set: " + e.getMessage());
                result.put("error", e.getClass().getSimpleName());
            }
        }

        return result;
    }

    /**
     * Creates a set with multiple values in a single operation.
     *
     * <p>This method demonstrates how to create a set with multiple values
     * in a single update operation.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param setAttributeName The name of the set attribute
     * @param setValues The values to include in the set
     * @param isNumberSet Whether to create a number set (true) or string set
(false)
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse createSetWithValues(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        Set<?> setValues,
        boolean isNumberSet) {

        AttributeValue setValue;

        if (isNumberSet) {
            // Convert numbers to strings for DynamoDB
```

```
            Set<String> stringValues = new HashSet<>();
            for (Object value : setValues) {
                if (value instanceof Number) {
                    stringValues.add(value.toString());
                } else {
                    throw new IllegalArgumentException("Values must be numbers for a
number set");
                }
            }

            setValue = AttributeValue.builder().ns(stringValues).build();
        } else {
            // Convert objects to strings for DynamoDB
            Set<String> stringValues = new HashSet<>();
            for (Object value : setValues) {
                stringValues.add(value.toString());
            }

            setValue = AttributeValue.builder().ss(stringValues).build();
        }

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #setAttr = :setValue")
            .expressionAttributeNames(Map.of("#setAttr", setAttributeName))
            .expressionAttributeValues(Map.of(":setValue", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Gets the current value of a set attribute.
     *
     * <p>Helper method to retrieve the current value of a set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @param setAttributeName The name of the set attribute
```

```
     * @return The set attribute value or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static AttributeValue getSetAttribute(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, String setAttributeName) {

        // Define the get parameters
        GetItemRequest request = GetItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .projectionExpression(setAttributeName)
            .build();

        try {
            // Perform the get operation
            GetItemResponse response = dynamoDbClient.getItem(request);

            // Return the set attribute if it exists, otherwise null
            if (response.item() != null &&
response.item().containsKey(setAttributeName)) {
                return response.item().get(setAttributeName);
            }

            return null;
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to get set attribute: " + e.getMessage())
                .cause(e)
                .build();
        }
    }
```

를 사용한 설정 작업 사용의 예입니다 AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating set operations in DynamoDB");
```

```
        try {
            // Example 1: Create a string set with multiple values
            System.out.println("\nExample 1: Creating a string set with multiple
values");
            Set<String> tags = new HashSet<>();
            tags.add("Electronics");
            tags.add("Gadget");
            tags.add("Smartphone");

            UpdateItemResponse createResponse = createSetWithValues(
                dynamoDbClient, tableName, key, "Tags", tags, false // Not a number
set
                );

            System.out.println("Created set attribute: " +
createResponse.attributes());

            // Example 2: Add values to a string set
            System.out.println("\nExample 2: Adding values to a string set");
            Set<String> additionalTags = new HashSet<>();
            additionalTags.add("Mobile");
            additionalTags.add("Wireless");

            UpdateItemResponse addResponse = addToStringSet(dynamoDbClient,
tableName, key, "Tags", additionalTags);

            System.out.println("Updated set attribute: " +
addResponse.attributes());

            // Example 3: Create a number set with multiple values
            System.out.println("\nExample 3: Creating a number set with multiple
values");
            Set<Number> ratings = new HashSet<>();
            ratings.add(4);
            ratings.add(5);
            ratings.add(4.5);

            UpdateItemResponse createNumberSetResponse = createSetWithValues(
                dynamoDbClient, tableName, key, "Ratings", ratings, true // Is a
number set
                );
```

```
            System.out.println("Created number set attribute: " +
createNumberSetResponse.attributes());

            // Example 4: Add values to a number set
            System.out.println("\nExample 4: Adding values to a number set");
            Set<Number> additionalRatings = new HashSet<>();
            additionalRatings.add(3.5);
            additionalRatings.add(4.2);

            UpdateItemResponse addNumberResponse =
                addToNumberSet(dynamoDbClient, tableName, key, "Ratings",
additionalRatings);

            System.out.println("Updated number set attribute: " +
addNumberResponse.attributes());

            // Example 5: Remove values from a set
            System.out.println("\nExample 5: Removing values from a set");
            Set<String> tagsToRemove = new HashSet<>();
            tagsToRemove.add("Gadget");

            UpdateItemResponse removeResponse = removeFromSet(
                dynamoDbClient, tableName, key, "Tags", tagsToRemove, false // Not a
number set
                );

            System.out.println("Updated set after removal: " +
removeResponse.attributes());

            // Example 6: Check if a value exists in a set
            System.out.println("\nExample 6: Checking if a value exists in a set");
            Map<String, Object> checkResult = checkIfValueInSet(dynamoDbClient,
tableName, key, "Tags", "Electronics");

            System.out.println("Check result: " + checkResult.get("message"));

            // Example 7: Get the current value of a set attribute
            System.out.println("\nExample 7: Getting the current value of a set
attribute");
            AttributeValue currentStringSet = getSetAttribute(dynamoDbClient,
tableName, key, "Tags");

            if (currentStringSet != null && currentStringSet.ss() != null) {
```

```
                System.out.println("Current string set values: " +
currentStringSet.ss());
            } else {
                System.out.println("String set attribute not found");
            }

            AttributeValue currentNumberSet = getSetAttribute(dynamoDbClient,
tableName, key, "Ratings");

            if (currentNumberSet != null && currentNumberSet.ns() != null) {
                System.out.println("Current number set values: " +
currentNumberSet.ns());
            } else {
                System.out.println("Number set attribute not found");
            }

            // Explain set operations
            System.out.println("\nKey points about DynamoDB set operations:");
            System.out.println(
                "1. DynamoDB supports three set types: string sets (SS), number sets
(NS), and binary sets (BS)");
            System.out.println("2. Sets can only contain elements of the same
type");
            System.out.println("3. Use ADD to add elements to a set");
            System.out.println("4. Use DELETE to remove elements from a set");
            System.out.println("5. Sets automatically remove duplicate values");
            System.out.println("6. Sets are unordered collections");
            System.out.println("7. Use the contains function to check if a value
exists in a set");
            System.out.println("8. You can create a set with multiple values in a
single operation");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateItem을 참조하세요.

PartiQL 문 배치를 사용하여 테이블 쿼리

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 여러 SELECT 문을 실행하여 항목 배치를 가져옵니다.
- 여러 INSERT 문을 실행하여 항목 배치를 추가합니다.
- 여러 UPDATE 문을 실행하여 항목 배치를 업데이트합니다.
- 여러 DELETE 문을 실행하여 항목 배치를 삭제합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);

        // Update multiple movies by using the BatchExecute statement.
        String title1 = "Star Wars";
        int year1 = 1977;
        String title2 = "Wizard of Oz";
        int year2 = 1939;

        System.out.println("Query two movies.");
```

```
        getBatch(ddb, tableName, title1, title2, year1, year2);

        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 boolean getBatch(DynamoDbClient ddb, String tableName, String
title1, String title2, int year1, int year2) {
        String getBatch = "SELECT * FROM " + tableName + " WHERE title = ? AND year
= ?";

        List<BatchStatementRequest> statements = new ArrayList<>();
        statements.add(BatchStatementRequest.builder()
            .statement(getBatch)
            .parameters(AttributeValue.builder().s(title1).build(),
                AttributeValue.builder().n(String.valueOf(year1)).build())
            .build());
        statements.add(BatchStatementRequest.builder()
            .statement(getBatch)
            .parameters(AttributeValue.builder().s(title2).build(),
                AttributeValue.builder().n(String.valueOf(year2)).build())
            .build());

        BatchExecuteStatementRequest batchExecuteStatementRequest =
BatchExecuteStatementRequest.builder()
            .statements(statements)
            .build();

        try {
            BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchExecuteStatementRequest);
            if (!response.responses().isEmpty()) {
                response.responses().forEach(r -> {
                    System.out.println(r.item().get("title") + "\\t" +
r.item().get("year"));
                });
                return true;
```

```
            } else {
                System.out.println("Couldn't find either " + title1 + " or " +
title2 + ".");
                return false;
            }
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            return false;
        }
    }

    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)
```

```
            .billingMode(BillingMode.PAY_PER_REQUEST) //  DynamoDB automatically
scales based on traffic.
            .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("1977")
                .build();

            AttributeValue att2 = AttributeValue.builder()
                .s("Star Wars")
                .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("1939")
                .build();

            AttributeValue attMovie2A = AttributeValue.builder()
                .s("Wizard of Oz")
                .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);
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 BatchExecuteStatement를 참조하세요.

PartiQL을 사용하여 테이블 쿼리

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- SELECT 문을 실행하여 항목을 가져옵니다.
- INSERT 문을 실행하여 항목을 추가합니다.
- UPDATE 문을 실행하여 항목을 업데이트합니다.
- DELETE 문을 실행하여 항목을 삭제합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
public class ScenarioPartiQ {
    public static void main(String[] args) throws IOException {
        String fileName = "../../../resources/sample_files/movies.json";
        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)
            .billingMode(BillingMode.PAY_PER_REQUEST) //Scales based on traffic.
            .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());
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ExecuteStatement를 참조하세요.

글로벌 보조 인덱스를 사용하여 테이블 쿼리

다음 코드 예제에서는 글로벌 보조 인덱스를 사용하여 테이블을 쿼리하는 방법을 보여줍니다.

- 프라이머리 키를 사용하여 DynamoDB 테이블을 쿼리합니다.
- 글로벌 보조 인덱스(GSI)에서 대체 액세스 패턴을 쿼리합니다.
- 테이블 쿼리와 GSI 쿼리를 비교합니다.

SDK for Java 2.x

기본 키와를 사용하는 글로벌 보조 인덱스(GSI)를 사용하여 DynamoDB 테이블을 쿼리합니다 AWS SDK for Java 2.x.

```
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 java.util.HashMap;
import java.util.Map;

    public QueryResponse queryTable(
        final String tableName, final String partitionKeyName, final String
 partitionKeyValue) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
```

```
        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query on base table successful. Found " +
response.count() + " items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw new DynamoDbQueryException("Table not found: " + tableName, e);
        } catch (DynamoDbException e) {
            System.err.println("Error querying base table: " + e.getMessage());
            throw new DynamoDbQueryException("Failed to execute query on base
table", e);
        }
    }

    /**
     * Queries a DynamoDB Global Secondary Index (GSI) by partition key.
     *
     * @param tableName         The name of the DynamoDB table
     * @param indexName         The name of the GSI
     * @param partitionKeyName  The name of the GSI partition key attribute
     * @param partitionKeyValue The value of the GSI partition key to query
     * @return The query response from DynamoDB
```

```
     * @throws ResourceNotFoundException if the table or index doesn't exist
     * @throws DynamoDbException if the query fails
     */
    public QueryResponse queryGlobalSecondaryIndex(
        final String tableName, final String indexName, final String
partitionKeyName, final String partitionKeyValue) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Index name", indexName);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_IK,
partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_IK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .indexName(indexName)
            .keyConditionExpression(GSI_KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query on GSI successful. Found " + response.count()
+ " items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format(
                "Error: The Amazon DynamoDB table \"%s\" or index \"%s\" can't be
found.\n", tableName, indexName);
            throw new DynamoDbQueryException("Table or index not found: " +
tableName + "/" + indexName, e);
        } catch (DynamoDbException e) {
```

```
            System.err.println("Error querying GSI: " + e.getMessage());
            throw new DynamoDbQueryException("Failed to execute query on GSI", e);
        }
    }
```

테이블 직접 쿼리와 GSI 쿼리를 비교합니다 AWS SDK for Java 2.x.

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <basePartitionKeyName> <basePartitionKeyValue>
<gsiName> <gsiPartitionKeyName> <gsiPartitionKeyValue> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    basePartitionKeyName - The name of the base table partition key
attribute.
                    basePartitionKeyValue - The value of the base table partition
key to query.
                    gsiName - The name of the Global Secondary Index.
                    gsiPartitionKeyName - The name of the GSI partition key
attribute.
                    gsiPartitionKeyValue - The value of the GSI partition key to
query.
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 6) {
            System.out.println(usage);
            System.exit(1);
        }

        final String tableName = args[0];
        final String basePartitionKeyName = args[1];
        final String basePartitionKeyValue = args[2];
        final String gsiName = args[3];
        final String gsiPartitionKeyName = args[4];
        final String gsiPartitionKeyValue = args[5];
        final Region region = args.length > 6 ? Region.of(args[6]) :
Region.US_EAST_1;
```

```
        try (DynamoDbClient ddb = DynamoDbClient.builder().region(region).build()) {
            final QueryTableAndGSI queryHelper = new QueryTableAndGSI(ddb);

            // Query the base table
            System.out.println("Querying base table where " + basePartitionKeyName +
" = " + basePartitionKeyValue);
            final QueryResponse tableResponse =
                queryHelper.queryTable(tableName, basePartitionKeyName,
basePartitionKeyValue);

            System.out.println("Found " + tableResponse.count() + " items in base
table:");
            tableResponse.items().forEach(item -> System.out.println(item));

            // Query the GSI
            System.out.println(
                "\nQuerying GSI '" + gsiName + "' where " + gsiPartitionKeyName + "
= " + gsiPartitionKeyValue);
            final QueryResponse gsiResponse =
                queryHelper.queryGlobalSecondaryIndex(tableName, gsiName,
gsiPartitionKeyName, gsiPartitionKeyValue);

            System.out.println("Found " + gsiResponse.count() + " items in GSI:");
            gsiResponse.items().forEach(item -> System.out.println(item));

            // Explain the differences between querying a table and a GSI
            System.out.println("\nKey differences between querying a table and a
GSI:");
            System.out.println("1. When querying a GSI, you must specify the
indexName parameter");
            System.out.println("2. GSIs may not contain all attributes from the base
table (projection)");
            System.out.println("3. GSIs consume read capacity units from the GSI's
capacity, not the base table's");
            System.out.println("4. GSIs may have eventually consistent data (cannot
use ConsistentRead=true)");

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table or index not found: " + e.getMessage());
            System.exit(1);
        } catch (DynamoDbException e) {
```

```
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Query를 참조하세요.

begins_with 조건을 사용하여 테이블 쿼리

다음 코드 예제에서는 begins_with 조건을 사용하여 테이블을 쿼리하는 방법을 보여줍니다.

- 키 조건 표현식에서 begins_with 함수를 사용합니다.
- 정렬 키의 접두사 패턴을 기준으로 항목을 필터링합니다.

SDK for Java 2.x

AWS SDK for Java 2.x로 정렬 키에 begins_with 조건을 사용하여 DynamoDB 테이블을 쿼리합니다.

```
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 java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithBeginsWithCondition(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String sortKeyName,
        final String sortKeyPrefix) {
```

```
        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Sort key name", sortKeyName);
        CodeSampleUtils.validateStringParameter("Sort key prefix", sortKeyPrefix);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_SK, sortKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_SK_PREFIX,
            AttributeValue.builder().s(sortKeyPrefix).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query with begins_with condition successful.
Found {0} items", response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with begins_with condition",
e);
            throw e;
        }
    }
```

AWS SDK for Java 2.x로 다양한 접두사 길이의 begins_with를 사용하는 방법을 보여줍니다.

```
   public static void main(String[] args) {
       try {
           CodeSampleUtils.BeginsWithQueryConfig config =
CodeSampleUtils.BeginsWithQueryConfig.fromArgs(args);
           LOGGER.log(Level.INFO, "Querying items where {0} = {1} and {2} begins
with ''{3}''", new Object[] {
               config.getPartitionKeyName(),
               config.getPartitionKeyValue(),
               config.getSortKeyName(),
               config.getSortKeyPrefix()
           });

           // Using the builder pattern to create and execute the query
           final QueryResponse response = new BeginsWithQueryBuilder()
               .withTableName(config.getTableName())
               .withPartitionKeyName(config.getPartitionKeyName())
               .withPartitionKeyValue(config.getPartitionKeyValue())
               .withSortKeyName(config.getSortKeyName())
               .withSortKeyPrefix(config.getSortKeyPrefix())
               .withRegion(config.getRegion())
               .execute();

           // Process the results
           LOGGER.log(Level.INFO, "Found {0} items:", response.count());
           response.items().forEach(item -> LOGGER.info(item.toString()));

           // Demonstrate with a different prefix
           if (!config.getSortKeyPrefix().isEmpty()) {
               String shorterPrefix = config.getSortKeyPrefix()
                   .substring(0, Math.max(1, config.getSortKeyPrefix().length() /
2));
               LOGGER.log(Level.INFO, "\nNow querying with a shorter prefix:
''{0}''", shorterPrefix);

               final QueryResponse response2 = new BeginsWithQueryBuilder()
                   .withTableName(config.getTableName())
                   .withPartitionKeyName(config.getPartitionKeyName())
                   .withPartitionKeyValue(config.getPartitionKeyValue())
                   .withSortKeyName(config.getSortKeyName())
```

```
                    .withSortKeyPrefix(shorterPrefix)
                    .withRegion(config.getRegion())
                    .execute();

                LOGGER.log(Level.INFO, "Found {0} items with shorter prefix:",
response2.count());
                response2.items().forEach(item -> LOGGER.info(item.toString()));
            }
        } catch (IllegalArgumentException e) {
            LOGGER.log(Level.SEVERE, "Invalid input: {0}", e.getMessage());
            printUsage();
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found", e);
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "DynamoDB error", e);
        } catch (Exception e) {
            LOGGER.log(Level.SEVERE, "Unexpected error", e);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Query를 참조하세요.

날짜 범위를 사용하여 테이블 쿼리

다음 코드 예제에서는 정렬 키에 날짜 범위를 사용하여 테이블을 쿼리하는 방법을 보여줍니다.

- 특정 날짜 범위 내의 항목을 쿼리합니다.
- 날짜 형식 정렬 키에 비교 연산자를 사용합니다.

SDK for Java 2.x

DynamoDB 테이블에서 날짜 범위 내의 항목을 쿼리합니다 AWS SDK for Java 2.x.

```
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 java.time.LocalDate;
import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithDateRange(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final LocalDate startDate,
        final LocalDate endDate) {

        // Focus on query logic, assuming parameters are valid
        if (startDate == null || endDate == null) {
            throw new IllegalArgumentException("Start date and end date cannot be
 null");
        }

        if (endDate.isBefore(startDate)) {
            throw new IllegalArgumentException("End date must be after start date");
        }

        // Format dates as ISO strings for DynamoDB (using just the date part)
        final String formattedStartDate = startDate.toString();
        final String formattedEndDate = endDate.toString();

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_SK, dateKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_START_DATE,
            AttributeValue.builder().s(formattedStartDate).build());
        expressionAttributeValues.put(
```

```
            EXPRESSION_ATTRIBUTE_VALUE_END_DATE,
            AttributeValue.builder().s(formattedEndDate).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query by date range successful. Found {0}
items", response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying by date range: {0}",
e.getMessage());
            throw e;
        }
    }
```

날짜 범위 필터링을 사용하여 DynamoDB 테이블을 쿼리하는 방법을 보여줍니다.

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue> <dateKeyName>
<startDate> <endDate> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    dateKeyName - The name of the date attribute to filter on.
                    startDate - The start date for the range query (YYYY-MM-DD).
                    endDate - The end date for the range query (YYYY-MM-DD).
```

```
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 6) {
            System.out.println(usage);
            System.exit(1);
        }

        try {
            // Parse command line arguments into a config object
            CodeSampleUtils.DateRangeQueryConfig config =
CodeSampleUtils.DateRangeQueryConfig.fromArgs(args);

            LOGGER.log(
                Level.INFO, "Querying items from {0} to {1}", new Object[]
{config.getStartDate(), config.getEndDate()
                });

            // Using the builder pattern to create and execute the query
            final QueryResponse response = new DateRangeQueryBuilder()
                .withTableName(config.getTableName())
                .withPartitionKeyName(config.getPartitionKeyName())
                .withPartitionKeyValue(config.getPartitionKeyValue())
                .withDateKeyName(config.getDateKeyName())
                .withStartDate(config.getStartDate())
                .withEndDate(config.getEndDate())
                .withRegion(config.getRegion())
                .execute();

            // Process the results
            LOGGER.log(Level.INFO, "Found {0} items:", response.count());
            response.items().forEach(item -> {
                LOGGER.info(item.toString());

                // Extract and display the date attribute for clarity
                if (item.containsKey(config.getDateKeyName())) {
                    LOGGER.log(
                        Level.INFO,
                        "  Date attribute: {0}",
                        item.get(config.getDateKeyName()).s());
                }
            });
```

```
            // Demonstrate with a different date range
            LocalDate narrowerStartDate = config.getStartDate().plusDays(1);
            LocalDate narrowerEndDate = config.getEndDate().minusDays(1);

            if (!narrowerStartDate.isAfter(narrowerEndDate)) {
                LOGGER.log(Level.INFO, "\nNow querying with a narrower date range:
{0} to {1}", new Object[] {
                    narrowerStartDate, narrowerEndDate
                });

                final QueryResponse response2 = new DateRangeQueryBuilder()
                    .withTableName(config.getTableName())
                    .withPartitionKeyName(config.getPartitionKeyName())
                    .withPartitionKeyValue(config.getPartitionKeyValue())
                    .withDateKeyName(config.getDateKeyName())
                    .withStartDate(narrowerStartDate)
                    .withEndDate(narrowerEndDate)
                    .withRegion(config.getRegion())
                    .execute();

                LOGGER.log(Level.INFO, "Found {0} items with narrower date range:",
response2.count());
                response2.items().forEach(item -> LOGGER.info(item.toString()));
            }

            LOGGER.info("\nNote: When storing dates in DynamoDB:");
            LOGGER.info("1. Use ISO format (YYYY-MM-DD) for lexicographical
ordering");
            LOGGER.info("2. Use the BETWEEN operator for inclusive date range
queries");
            LOGGER.info("3. Consider using ISO-8601 format for timestamps with time
components");

        } catch (IllegalArgumentException e) {
            LOGGER.log(Level.SEVERE, "Invalid input: {0}", e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", e.getMessage());
            System.exit(1);
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "DynamoDB error: {0}", e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            LOGGER.log(Level.SEVERE, "Unexpected error: {0}", e.getMessage());
```

```
            System.exit(1);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Query를 참조하세요.

복잡한 필터 표현식을 사용하여 테이블 쿼리

다음 코드 예제에서는 복잡한 필터 표현식을 사용하여 테이블을 쿼리하는 방법을 보여줍니다.

- 쿼리 결과에 복잡한 필터 표현식을 적용합니다.
- 논리적 연산자를 사용하여 여러 조건을 결합합니다.
- 키가 아닌 속성을 기준으로 항목을 필터링합니다.

SDK for Java 2.x

를 사용하여 복잡한 필터 표현식을 사용하여 DynamoDB 테이블을 쿼리합니다 AWS SDK for Java 2.x.

```
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 java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithComplexFilter(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String statusAttrName,
        final String activeStatus,
        final String pendingStatus,
        final String priceAttrName,
```

```
        final double minPrice,
        final double maxPrice,
        final String categoryAttrName) {

        // Validate parameters
        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Status attribute name",
statusAttrName);
        CodeSampleUtils.validateStringParameter("Active status", activeStatus);
        CodeSampleUtils.validateStringParameter("Pending status", pendingStatus);
        CodeSampleUtils.validateStringParameter("Price attribute name",
priceAttrName);
        CodeSampleUtils.validateStringParameter("Category attribute name",
categoryAttrName);
        CodeSampleUtils.validateNumericRange("Minimum price", minPrice, 0.0,
Double.MAX_VALUE);
        CodeSampleUtils.validateNumericRange("Maximum price", maxPrice, minPrice,
Double.MAX_VALUE);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#pk", partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_STATUS,
statusAttrName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PRICE,
priceAttrName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_CATEGORY,
categoryAttrName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            ":pkValue", AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_ACTIVE,
            AttributeValue.builder().s(activeStatus).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PENDING,
            AttributeValue.builder().s(pendingStatus).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_MIN_PRICE,
            AttributeValue.builder().n(String.valueOf(minPrice)).build());
```

```
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_MAX_PRICE,
            AttributeValue.builder().n(String.valueOf(maxPrice)).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(FILTER_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        return dynamoDbClient.query(queryRequest);
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Query를 참조하세요.

동적 필터 표현식을 사용하여 테이블 쿼리

다음 코드 예제에서는 동적 필터 표현식을 사용하여 테이블을 쿼리하는 방법을 보여줍니다.

- 런타임에 동적으로 필터 표현식을 작성합니다.
- 사용자 입력 또는 애플리케이션 상태를 기반으로 필터 조건을 구성합니다.
- 조건부로 필터 기준을 추가하거나 제거합니다.

SDK for Java 2.x

를 사용하여 동적으로 구성된 필터 표현식을 사용하여 DynamoDB 테이블을 쿼리합니다 AWS SDK for Java 2.x.

```
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 java.util.HashMap;
```

```
import java.util.Map;

    public static QueryResponse queryWithDynamicFilter(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final Map<String, Object> filterCriteria,
        final Region region,
        final DynamoDbClient dynamoDbClient) {

        validateParameters(tableName, partitionKeyName, partitionKeyValue,
 filterCriteria);

        DynamoDbClient ddbClient = dynamoDbClient;
        boolean shouldClose = false;

        try {
            if (ddbClient == null) {
                ddbClient = createClient(region);
                shouldClose = true;
            }

            final QueryWithDynamicFilter queryHelper = new
 QueryWithDynamicFilter(ddbClient);
            return queryHelper.queryWithDynamicFilter(tableName, partitionKeyName,
 partitionKeyValue, filterCriteria);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Failed to execute dynamic filter query: " +
 e.getMessage());
            throw e;
        } catch (Exception e) {
            System.err.println("Unexpected error during query: " + e.getMessage());
            throw e;
        } finally {
            if (shouldClose && ddbClient != null) {
                ddbClient.close();
            }
        }
    }
```

동적 필터 표현식을와 함께 사용하는 방법을 보여줍니다 AWS SDK for Java 2.x.

```
   public static void main(String[] args) {
       final String usage =
           """
               Usage:
                   <tableName> <partitionKeyName> <partitionKeyValue>
<filterAttrName> <filterAttrValue> [region]
               Where:
                   tableName - The Amazon DynamoDB table to query.
                   partitionKeyName - The name of the partition key attribute.
                   partitionKeyValue - The value of the partition key to query.
                   filterAttrName - The name of the attribute to filter on.
                   filterAttrValue - The value to filter by.
                   region (optional) - The AWS region where the table exists.
(Default: us-east-1)
               """;

       if (args.length < 5) {
           System.out.println(usage);
           System.exit(1);
       }

       final String tableName = args[0];
       final String partitionKeyName = args[1];
       final String partitionKeyValue = args[2];
       final String filterAttrName = args[3];
       final String filterAttrValue = args[4];
       final Region region = args.length > 5 ? Region.of(args[5]) :
Region.US_EAST_1;

       System.out.println("Querying items with dynamic filter: " + filterAttrName +
" = " + filterAttrValue);

       try {
           // Using the builder pattern to create and execute the query
           final QueryResponse response = new DynamicFilterQueryBuilder()
               .withTableName(tableName)
               .withPartitionKeyName(partitionKeyName)
               .withPartitionKeyValue(partitionKeyValue)
               .withFilterCriterion(filterAttrName, filterAttrValue)
               .withRegion(region)
               .execute();
```

```
            // Process the results
            System.out.println("Found " + response.count() + " items:");
            response.items().forEach(item -> System.out.println(item));

            // Demonstrate multiple filter criteria
            System.out.println("\nNow querying with multiple filter criteria:");

            Map<String, Object> multipleFilters = new HashMap<>();
            multipleFilters.put(filterAttrName, filterAttrValue);
            multipleFilters.put("status", "active");

            final QueryResponse multiFilterResponse = new
DynamicFilterQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withFilterCriteria(multipleFilters)
                .withRegion(region)
                .execute();

            System.out.println("Found " + multiFilterResponse.count() + " items with
multiple filters:");
            multiFilterResponse.items().forEach(item -> System.out.println(item));

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Query를 참조하세요.

필터 표현식 및 제한을 사용하여 테이블 쿼리

다음 코드 예제에서는 필터 표현식 및 제한을 사용하여 테이블을 쿼리하는 방법을 보여줍니다.

- 평가되는 항목에 제한을 두어 쿼리 결과에 필터 표현식을 적용합니다.
- 제한이 필터링된 쿼리 결과에 미치는 영향을 이해합니다.
- 쿼리에서 처리되는 최대 항목 수를 제어합니다.

SDK for Java 2.x

를 사용하여 필터 표현식 및 제한으로 DynamoDB 테이블을 쿼리합니다 AWS SDK for Java 2.x.

```
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 java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithFilterAndLimit(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String filterAttrName,
        final String filterAttrValue,
        final int limit) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Filter attribute name",
 filterAttrName);
        CodeSampleUtils.validateStringParameter("Filter attribute value",
 filterAttrValue);
        CodeSampleUtils.validatePositiveInteger("Limit", limit);

        // Create expression attribute names for the column names
```

```
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_FILTER,
filterAttrName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_FILTER,
            AttributeValue.builder().s(filterAttrValue).build());

        // Create the filter expression
        final String filterExpression = "#filterAttr = :filterValue";

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(filterExpression)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .limit(limit)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query with filter and limit successful. Found
{0} items", response.count());
            LOGGER.log(
                Level.INFO, "ScannedCount: {0} (total items evaluated before
filtering)", response.scannedCount());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with filter and limit: {0}",
e.getMessage());
            throw e;
```

```
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Query를 참조하세요.

중첩 속성을 사용하여 테이블 쿼리

다음 코드 예제에서는 중첩 속성을 사용하여 테이블을 쿼리하는 방법을 보여줍니다.

- DynamoDB 항목의 중첩 속성을 기준으로 액세스하고 필터링합니다.
- 문서 경로 표현식을 사용하여 중첩된 요소를 참조합니다.

SDK for Java 2.x

를 사용하여 중첩 속성으로 DynamoDB 테이블을 쿼리합니다 AWS SDK for Java 2.x.

```
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 java.util.HashMap;
import java.util.Map;

    public QueryResponse queryWithNestedAttributes(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String nestedPath,
        final String nestedAttr,
        final String nestedValue) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Nested path", nestedPath);
        CodeSampleUtils.validateStringParameter("Nested attribute", nestedAttr);
        CodeSampleUtils.validateStringParameter("Nested value", nestedValue);
```

```
        // Split the nested path into components
        final String[] pathComponents = nestedPath.split("\\.");

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);

        // Build the nested attribute reference using document path notation
        final StringBuilder nestedAttributeRef = new StringBuilder();
        for (int i = 0; i < pathComponents.length; i++) {
            final String aliasName = "#n" + i;
            expressionAttributeNames.put(aliasName, pathComponents[i]);

            if (i > 0) {
                nestedAttributeRef.append(".");
            }
            nestedAttributeRef.append(aliasName);
        }

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_NESTED,
            AttributeValue.builder().s(nestedValue).build());

        // Create the filter expression using the nested attribute reference
        final String filterExpression = nestedAttributeRef + " = :nestedValue";

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(filterExpression)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
```

```
            System.out.println("Query with nested attribute filter successful. Found
" + response.count() + " items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with nested attribute filter: " +
e.getMessage());
            throw e;
        }
    }
```

중첩 속성을 사용하여 DynamoDB 테이블을 쿼리하는 방법을 보여줍니다.

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue> <nestedPath>
<nestedAttr> <nestedValue> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    nestedPath - The path to the nested map attribute (e.g.,
"address").
                    nestedAttr - The name of the nested attribute (e.g., "city").
                    nestedValue - The value to filter by (e.g., "Seattle").
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 6) {
            System.out.println(usage);
            System.exit(1);
        }

        final String tableName = args[0];
        final String partitionKeyName = args[1];
        final String partitionKeyValue = args[2];
```

```
        final String nestedPath = args[3];
        final String nestedAttr = args[4];
        final String nestedValue = args[5];
        final Region region = args.length > 6 ? Region.of(args[6]) :
Region.US_EAST_1;

        System.out.println("Querying items where " + partitionKeyName + " = " +
partitionKeyValue + " and " + nestedPath
            + "." + nestedAttr + " = " + nestedValue);

        try {
            // Using the builder pattern to create and execute the query
            final QueryResponse response = new NestedAttributeQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withNestedPath(nestedPath)
                .withNestedAttribute(nestedAttr)
                .withNestedValue(nestedValue)
                .withRegion(region)
                .execute();

            // Process the results
            System.out.println("Found " + response.count() + " items:");
            response.items().forEach(item -> {
                System.out.println(item);

                // Extract and display the nested attribute for clarity
                if (item.containsKey(nestedPath) && item.get(nestedPath).hasM()) {
                    Map<String, AttributeValue> nestedMap =
item.get(nestedPath).m();
                    if (nestedMap.containsKey(nestedAttr)) {
                        System.out.println("  Nested attribute " + nestedPath + "."
+ nestedAttr + ": "
                                + formatAttributeValue(nestedMap.get(nestedAttr)));
                    }
                }
            });

            System.out.println("\nNote: When working with nested attributes in
DynamoDB:");
            System.out.println("1. Use dot notation in filter expressions to access
nested attributes");
```

```
            System.out.println("2. Use expression attribute names for each component
of the path");
            System.out.println("3. Check if the nested attribute exists before
accessing it");

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Query를 참조하세요.

페이지 매김을 사용하여 테이블 쿼리

다음 코드 예제에서는 페이지 매김을 사용하여 테이블을 쿼리하는 방법을 보여줍니다.

- DynamoDB 쿼리 결과에 페이지 매김을 구현합니다.
- LastEvaluatedKey를 사용하여 후속 페이지를 검색합니다.
- Limit 파라미터를 사용하여 페이지당 항목 수를 제어합니다.

SDK for Java 2.x

를 사용하여 페이지 매김으로 DynamoDB 테이블을 쿼리합니다 AWS SDK for Java 2.x.

```
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 java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    public List<Map<String, AttributeValue>> queryWithPagination(
        final String tableName, final String partitionKeyName, final String
 partitionKeyValue, final int pageSize) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validatePositiveInteger("Page size", pageSize);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        QueryRequest.Builder queryRequestBuilder = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .limit(pageSize);

        // List to store all items from all pages
        final List<Map<String, AttributeValue>> allItems = new ArrayList<>();

        // Map to store the last evaluated key for pagination
        Map<String, AttributeValue> lastEvaluatedKey = null;
        int pageNumber = 1;

        try {
            do {
```

```
                // If we have a last evaluated key, use it for the next page
                if (lastEvaluatedKey != null) {
                    queryRequestBuilder.exclusiveStartKey(lastEvaluatedKey);
                }

                // Execute the query
                final QueryResponse response =
dynamoDbClient.query(queryRequestBuilder.build());

                // Process the current page of results
                final List<Map<String, AttributeValue>> pageItems =
response.items();
                allItems.addAll(pageItems);

                // Get the last evaluated key for the next page
                lastEvaluatedKey = response.lastEvaluatedKey();
                if (lastEvaluatedKey != null && lastEvaluatedKey.isEmpty()) {
                    lastEvaluatedKey = null;
                }

                System.out.println("Page " + pageNumber + ": Retrieved " +
pageItems.size() + " items (Running total: "
                    + allItems.size() + ")");

                pageNumber++;

            } while (lastEvaluatedKey != null);

            System.out.println("Query with pagination complete. Retrieved a total of
" + allItems.size()
                + " items across " + (pageNumber - 1) + " pages");

            return allItems;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with pagination: " + e.getMessage());
            throw e;
        }
    }
```

페이지 매김을 사용하여 DynamoDB 테이블을 쿼리하는 방법을 보여줍니다.

```
   public static void main(String[] args) {
       final String usage =
           """
               Usage:
                   <tableName> <partitionKeyName> <partitionKeyValue> [pageSize]
[region]
               Where:
                   tableName - The Amazon DynamoDB table to query.
                   partitionKeyName - The name of the partition key attribute.
                   partitionKeyValue - The value of the partition key to query.
                   pageSize (optional) - The maximum number of items to return per
page. (Default: 10)
                   region (optional) - The AWS region where the table exists.
(Default: us-east-1)
               """;

       if (args.length < 3) {
           System.out.println(usage);
           System.exit(1);
       }

       final String tableName = args[0];
       final String partitionKeyName = args[1];
       final String partitionKeyValue = args[2];
       final int pageSize = args.length > 3 ? Integer.parseInt(args[3]) : 10;
       final Region region = args.length > 4 ? Region.of(args[4]) :
Region.US_EAST_1;

       System.out.println("Querying items with pagination (page size: " + pageSize
+ ")");

       try {
           // Using the builder pattern to create and execute the query
           final List<Map<String, AttributeValue>> allItems = new
PaginationQueryBuilder()
               .withTableName(tableName)
               .withPartitionKeyName(partitionKeyName)
               .withPartitionKeyValue(partitionKeyValue)
               .withPageSize(pageSize)
               .withRegion(region)
               .executeWithPagination();
```

```
            // Process the results
            System.out.println("\nSummary: Retrieved a total of " + allItems.size()
+ " items");

            // Display the first few items as a sample
            final int sampleSize = Math.min(5, allItems.size());
            if (sampleSize > 0) {
                System.out.println("\nSample of retrieved items (first " +
sampleSize + "):");
                for (int i = 0; i < sampleSize; i++) {
                    System.out.println(allItems.get(i));
                }

                if (allItems.size() > sampleSize) {
                    System.out.println("... and " + (allItems.size() - sampleSize) +
" more items");
                }
            }
        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Query를 참조하세요.

강력하게 일관된 읽기를 사용하여 테이블 쿼리

다음 코드 예제에서는 강력하게 일관된 읽기를 사용하여 테이블을 쿼리하는 방법을 보여줍니다.

- DynamoDB 쿼리의 일관성 수준을 구성합니다.
- 강력하게 일관된 읽기를 사용하여 최신 데이터를 가져옵니다.

- 최종 일관성과 강력한 일관성의 장단점을 이해합니다.

SDK for Java 2.x

를 사용하여 구성 가능한 읽기 일관성으로 DynamoDB 테이블을 쿼리합니다 AWS SDK for Java 2.x.

```
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 java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithConsistentReads(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final boolean useConsistentRead) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
```

```
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .consistentRead(useConsistentRead)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query successful. Found {0} items",
response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with consistent reads", e);
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Query를 참조하세요.

TTL 항목에 대한 쿼리

다음 코드 예제에서는 TTL 항목을 쿼리하는 방법을 보여줍니다.

SDK for Java 2.x

필터링된 표현식을 쿼리하여를 사용하여 DynamoDB 테이블에서 TTL 항목을 수집합니다 AWS SDK for Java 2.x.

```
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 java.util.Map;
import java.util.Optional;
```

```
        final QueryRequest request = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(FILTER_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            final QueryResponse response = ddb.query(request);
            System.out.println("Query successful. Found " + response.count() + "
items that have not expired yet.");

            // Print each item
            response.items().forEach(item -> {
                System.out.println("Item: " + item);
            });

            return 0;
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Query를 참조하세요.

날짜 및 시간 패턴을 사용하여 테이블 쿼리

다음 코드 예제에서는 날짜 및 시간 패턴을 사용하여 테이블을 쿼리하는 방법을 보여줍니다.

- DynamoDB에 날짜/시간 값을 저장하고 쿼리합니다.
- 정렬 키를 사용하여 날짜 범위 쿼리를 구현합니다.
- 효과적인 쿼리를 위해 날짜 문자열의 형식을 지정합니다.

SDK for Java 2.x

에서 정렬 키의 날짜 범위를 사용하여 쿼리합니다 AWS SDK for Java 2.x.

```
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 java.time.LocalDate;
import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithDateRange(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final LocalDate startDate,
        final LocalDate endDate) {

        // Focus on query logic, assuming parameters are valid
        if (startDate == null || endDate == null) {
            throw new IllegalArgumentException("Start date and end date cannot be
 null");
        }

        if (endDate.isBefore(startDate)) {
            throw new IllegalArgumentException("End date must be after start date");
        }

        // Format dates as ISO strings for DynamoDB (using just the date part)
        final String formattedStartDate = startDate.toString();
        final String formattedEndDate = endDate.toString();

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);
```

```
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_SK, dateKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_START_DATE,
            AttributeValue.builder().s(formattedStartDate).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_END_DATE,
            AttributeValue.builder().s(formattedEndDate).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query by date range successful. Found {0}
 items", response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying by date range: {0}",
 e.getMessage());
            throw e;
        }
    }
```

에서 날짜-시간 변수를 사용하여 쿼리합니다 AWS SDK for Java 2.x.

```
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 java.time.Instant;
import java.time.LocalDateTime;
import java.time.ZoneOffset;
import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryWithDateTime(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final String startDate,
        final String endDate) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateDateRangeParameters(dateKeyName, startDate,
 endDate);
        CodeSampleUtils.validateDateFormat("Start date", startDate);
        CodeSampleUtils.validateDateFormat("End date", endDate);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);
        expressionAttributeNames.put("#dateKey", dateKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            ":startDate", AttributeValue.builder().s(startDate).build());
        expressionAttributeValues.put(
            ":endDate", AttributeValue.builder().s(endDate).build());

        // Create the query request
```

```
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query successful. Found " + response.count() + "
 items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
 found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with date range: " + e.getMessage());
            throw e;
        }
    }
```

AWS SDK for Java 2.x로 Unix 에포크 타임스탬프의 날짜 범위 내에서 쿼리합니다.

```
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 java.time.Instant;
import java.time.LocalDateTime;
import java.time.ZoneOffset;
import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryWithDateTimeEpoch(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
```

```
        final long startEpoch,
        final long endEpoch) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Date key name", dateKeyName);
        CodeSampleUtils.validateEpochTimestamp("Start epoch", startEpoch);
        CodeSampleUtils.validateEpochTimestamp("End epoch", endEpoch);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);
        expressionAttributeNames.put("#dateKey", dateKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            ":startDate",
AttributeValue.builder().n(String.valueOf(startEpoch)).build());
        expressionAttributeValues.put(
            ":endDate",
AttributeValue.builder().n(String.valueOf(endEpoch)).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query successful. Found " + response.count() + "
items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
```

```
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with epoch timestamps: " +
 e.getMessage());
            throw e;
        }
    }
```

LocalDateTime 객체를와 함께 사용하여 날짜 범위 내에서 쿼리합니다 AWS SDK for Java 2.x.

```
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 java.time.Instant;
import java.time.LocalDateTime;
import java.time.ZoneOffset;
import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryWithDateTimeLocalDateTime(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final LocalDateTime startDateTime,
        final LocalDateTime endDateTime) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Date key name", dateKeyName);
        if (startDateTime == null || endDateTime == null) {
            throw new IllegalArgumentException("Start and end LocalDateTime must not
 be null");
        }

        // Convert LocalDateTime to ISO-8601 strings in UTC with the correct format
        final String startDate =
 startDateTime.atZone(ZoneOffset.UTC).format(DATE_TIME_FORMATTER);
```

```
        final String endDate =
endDateTime.atZone(ZoneOffset.UTC).format(DATE_TIME_FORMATTER);

        return queryWithDateTime(tableName, partitionKeyName, partitionKeyValue,
dateKeyName, startDate, endDate);
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Query를 참조하세요.

업데이트 표현식 순서 이해

다음 코드 예제에서는 업데이트 표현식 순서를 이해하는 방법을 보여줍니다.

- DynamoDB가 업데이트 표현식을 처리하는 방법을 알아봅니다.
- 업데이트 표현식의 연산 순서를 이해합니다.
- 표현식 평가를 이해하여 예상치 못한 결과를 방지합니다.

SDK for Java 2.x

를 사용하여 업데이트 표현식 순서를 보여줍니다 AWS SDK for Java 2.x.

```
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.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ReturnValue;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.Map;

    /**
     * Demonstrates the effect of update expression order.
     *
     * <p>This method shows how the order of operations in an update expression
     * affects the result of the update.
     *
     * @param dynamoDbClient The DynamoDB client
```

```
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @return Map containing the results of different update orders
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Map<String, Object> demonstrateUpdateOrder(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key) {

        Map<String, Object> results = new HashMap<>();

        try {
            // Initialize the item with a counter
            UpdateItemRequest initRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET Counter = :zero, OldCounter = :zero")
                .expressionAttributeValues(
                    Map.of(":zero", AttributeValue.builder().n("0").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            dynamoDbClient.updateItem(initRequest);

            // Example 1: SET first, then ADD
            UpdateItemRequest setFirstRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET Counter = :value ADD OldCounter :increment")
                .expressionAttributeValues(Map.of(
                    ":value", AttributeValue.builder().n("10").build(),
                    ":increment", AttributeValue.builder().n("5").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse setFirstResponse =
dynamoDbClient.updateItem(setFirstRequest);
            results.put("setFirstResponse", setFirstResponse);

            // Reset the item
            dynamoDbClient.updateItem(initRequest);

            // Example 2: ADD first, then SET
            UpdateItemRequest addFirstRequest = UpdateItemRequest.builder()
```

```
                .tableName(tableName)
                .key(key)
                .updateExpression("ADD Counter :increment SET OldCounter = :value")
                .expressionAttributeValues(Map.of(
                    ":value", AttributeValue.builder().n("10").build(),
                    ":increment", AttributeValue.builder().n("5").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse addFirstResponse =
dynamoDbClient.updateItem(addFirstRequest);
            results.put("addFirstResponse", addFirstResponse);

            // Reset the item
            dynamoDbClient.updateItem(initRequest);

            // Example 3: SET with multiple attributes
            UpdateItemRequest multiSetRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET Counter = :value, OldCounter = Counter")
                .expressionAttributeValues(
                    Map.of(":value", AttributeValue.builder().n("10").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse multiSetResponse =
dynamoDbClient.updateItem(multiSetRequest);
            results.put("multiSetResponse", multiSetResponse);

            // Reset the item
            dynamoDbClient.updateItem(initRequest);

            // Example 4: SET with expression using the same attribute
            UpdateItemRequest selfReferenceRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET Counter = Counter + :increment, OldCounter =
Counter")
                .expressionAttributeValues(
                    Map.of(":increment", AttributeValue.builder().n("5").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();
```

```
            UpdateItemResponse selfReferenceResponse =
dynamoDbClient.updateItem(selfReferenceRequest);
            results.put("selfReferenceResponse", selfReferenceResponse);

            results.put("success", true);

        } catch (DynamoDbException e) {
            results.put("success", false);
            results.put("error", e.getMessage());
        }

        return results;
    }

    /**
     * Updates an item with SET first, then REMOVE.
     *
     * <p>This method demonstrates updating an item with SET operation first,
     * followed by a REMOVE operation.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param attributeToSet The attribute to set
     * @param setValue The value to set
     * @param attributeToRemove The attribute to remove
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateWithSetFirst(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String attributeToSet,
        AttributeValue setValue,
        String attributeToRemove) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #setAttr = :setValue REMOVE #removeAttr")
            .expressionAttributeNames(Map.of(
                "#setAttr", attributeToSet,
```

```
                "#removeAttr", attributeToRemove))
            .expressionAttributeValues(Map.of(":setValue", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        try {
            return dynamoDbClient.updateItem(request);
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to update item with SET first: " + e.getMessage())
                .cause(e)
                .build();
        }
    }

    /**
     * Updates an item with REMOVE first, then SET.
     *
     * <p>This method demonstrates updating an item with REMOVE operation first,
     * followed by a SET operation.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param attributeToSet The attribute to set
     * @param setValue The value to set
     * @param attributeToRemove The attribute to remove
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateWithRemoveFirst(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String attributeToSet,
        AttributeValue setValue,
        String attributeToRemove) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("REMOVE #removeAttr SET #setAttr = :setValue")
```

```
            .expressionAttributeNames(Map.of(
                "#setAttr", attributeToSet,
                "#removeAttr", attributeToRemove))
            .expressionAttributeValues(Map.of(":setValue", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        try {
            return dynamoDbClient.updateItem(request);
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to update item with REMOVE first: " +
e.getMessage())
                .cause(e)
                .build();
        }
    }

    /**
     * Updates an item with all operation types in a specific order.
     *
     * <p>This method demonstrates using all operation types (SET, REMOVE, ADD,
DELETE)
     * in a specific order in a single update expression.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateWithAllOperationTypes(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #stringAttr = :stringVal, #mapAttr.#nestedAttr
= :nestedVal " + "REMOVE #oldAttr "
                + "ADD #counterAttr :increment "
                + "DELETE #stringSetAttr :stringSetVal")
```

```
            .expressionAttributeNames(Map.of(
                "#stringAttr", "StringAttribute",
                "#mapAttr", "MapAttribute",
                "#nestedAttr", "NestedAttribute",
                "#oldAttr", "OldAttribute",
                "#counterAttr", "CounterAttribute",
                "#stringSetAttr", "StringSetAttribute"))
            .expressionAttributeValues(Map.of(
                ":stringVal", AttributeValue.builder().s("New Value").build(),
                ":nestedVal", AttributeValue.builder().s("Nested Value").build(),
                ":increment", AttributeValue.builder().n("1").build(),
                ":stringSetVal", AttributeValue.builder().ss("Value1").build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        try {
            return dynamoDbClient.updateItem(request);
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to update item with all operation types: " +
e.getMessage())
                .cause(e)
                .build();
        }
    }

    /**
     * Gets the current state of an item.
     *
     * <p>Helper method to retrieve the current state of an item.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @return The item or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Map<String, AttributeValue> getItem(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key) {

        // Define the get parameters
        GetItemRequest request =
```

```
            GetItemRequest.builder().tableName(tableName).key(key).build();

        // Perform the get operation
        try {
            GetItemResponse response = dynamoDbClient.getItem(request);

            // Return the item if it exists, otherwise null
            return response.item();
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to get item: " + e.getMessage())
                .cause(e)
                .build();
        }
    }
```

를 사용한 업데이트 표현식 순서 사용의 예입니다 AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating update expression order in DynamoDB");

        try {
            // Example 1: Demonstrate update order effects
            System.out.println("\nExample 1: Demonstrating update order effects");
            Map<String, Object> orderResults =
demonstrateUpdateOrder(dynamoDbClient, tableName, key);

            if ((boolean) orderResults.get("success")) {
                System.out.println("SET first, then ADD:");
                System.out.println("  " + orderResults.get("setFirstResponse"));

                System.out.println("ADD first, then SET:");
                System.out.println("  " + orderResults.get("addFirstResponse"));

                System.out.println("SET with multiple attributes:");
                System.out.println("  " + orderResults.get("multiSetResponse"));
```

```
                System.out.println("SET with self-reference:");
                System.out.println("  " +
orderResults.get("selfReferenceResponse"));
            } else {
                System.out.println("Error: " + orderResults.get("error"));
            }

            // Example 2: Update with SET first, then REMOVE
            System.out.println("\nExample 2: Update with SET first, then REMOVE");
            UpdateItemResponse setFirstResponse = updateWithSetFirst(
                dynamoDbClient,
                tableName,
                key,
                "Status",
                AttributeValue.builder().s("Active").build(),
                "OldStatus");

            System.out.println("Updated attributes: " +
setFirstResponse.attributes());

            // Example 3: Update with REMOVE first, then SET
            System.out.println("\nExample 3: Update with REMOVE first, then SET");
            UpdateItemResponse removeFirstResponse = updateWithRemoveFirst(
                dynamoDbClient,
                tableName,
                key,
                "Status",
                AttributeValue.builder().s("Inactive").build(),
                "OldStatus");

            System.out.println("Updated attributes: " +
removeFirstResponse.attributes());

            // Example 4: Update with all operation types
            System.out.println("\nExample 4: Update with all operation types");
            UpdateItemResponse allOpsResponse =
updateWithAllOperationTypes(dynamoDbClient, tableName, key);

            System.out.println("Updated attributes: " +
allOpsResponse.attributes());

            // Example 5: Get the current state of the item
            System.out.println("\nExample 5: Current state of the item");
```

```
            Map<String, AttributeValue> item = getItem(dynamoDbClient, tableName,
key);

            if (item != null) {
                System.out.println("Item: " + item);
            } else {
                System.out.println("Item not found");
            }

            // Explain update expression order
            System.out.println("\nKey points about update expression order in
DynamoDB:");
            System.out.println("1. Update expressions are processed in this order:
SET, REMOVE, ADD, DELETE");
            System.out.println("2. Within each clause, operations are processed from
left to right");
            System.out.println("3. SET operations use the item state before any
updates in the expression");
            System.out.println("4. When an attribute is referenced multiple times,
the first operation wins");
            System.out.println("5. To reference a new value, split the update into
multiple operations");
            System.out.println("6. The order of clauses in the expression doesn't
change the evaluation order");
            System.out.println("7. For complex updates, consider using multiple
separate update operations");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateItem을 참조하세요.

테이블의 웜 처리량 설정 업데이트

다음 코드 예제에서는 테이블의 웜 처리량 설정을 업데이트하는 방법을 보여줍니다.

SDK for Java 2.x

AWS SDK for Java 2.x를 사용하여 기존 DynamoDB 테이블에서 웜 처리량 설정을 업데이트합니다.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GlobalSecondaryIndexUpdate;
import
 software.amazon.awssdk.services.dynamodb.model.UpdateGlobalSecondaryIndexAction;
import software.amazon.awssdk.services.dynamodb.model.UpdateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.WarmThroughput;

    public static WarmThroughput buildWarmThroughput(final Long readUnitsPerSecond,
 final Long writeUnitsPerSecond) {
        return WarmThroughput.builder()
            .readUnitsPerSecond(readUnitsPerSecond)
            .writeUnitsPerSecond(writeUnitsPerSecond)
            .build();
    }

    /**
     * Updates a DynamoDB table with warm throughput settings for both the table and
 a global secondary index.
     *
     * @param ddb The DynamoDB client
     * @param tableName The name of the table to update
     * @param tableReadUnitsPerSecond Read units per second for the table
     * @param tableWriteUnitsPerSecond Write units per second for the table
     * @param globalSecondaryIndexName The name of the global secondary index to
 update
     * @param globalSecondaryIndexReadUnitsPerSecond Read units per second for the
 GSI
     * @param globalSecondaryIndexWriteUnitsPerSecond Write units per second for the
 GSI
     */
    public static void updateDynamoDBTable(
        final DynamoDbClient ddb,
        final String tableName,
        final Long tableReadUnitsPerSecond,
        final Long tableWriteUnitsPerSecond,
        final String globalSecondaryIndexName,
        final Long globalSecondaryIndexReadUnitsPerSecond,
        final Long globalSecondaryIndexWriteUnitsPerSecond) {
```

```
        final WarmThroughput tableWarmThroughput =
            buildWarmThroughput(tableReadUnitsPerSecond, tableWriteUnitsPerSecond);
        final WarmThroughput gsiWarmThroughput =
            buildWarmThroughput(globalSecondaryIndexReadUnitsPerSecond,
globalSecondaryIndexWriteUnitsPerSecond);

        final GlobalSecondaryIndexUpdate globalSecondaryIndexUpdate =
GlobalSecondaryIndexUpdate.builder()
            .update(UpdateGlobalSecondaryIndexAction.builder()
                .indexName(globalSecondaryIndexName)
                .warmThroughput(gsiWarmThroughput)
                .build())
            .build();

        final UpdateTableRequest request = UpdateTableRequest.builder()
            .tableName(tableName)
            .globalSecondaryIndexUpdates(globalSecondaryIndexUpdate)
            .warmThroughput(tableWarmThroughput)
            .build();

        try {
            ddb.updateTable(request);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }

        System.out.println(SUCCESS_MESSAGE);
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateTable을 참조하세요.

항목의 TTL 업데이트

다음 코드 예제에서는 항목의 TTL을 업데이트하는 방법을 보여줍니다.

SDK for Java 2.x

테이블의 기존 DynamoDB 항목에서 TTL을 업데이트합니다.

```
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 java.util.HashMap;
import java.util.Map;
import java.util.Optional;

    public UpdateItemResponse updateItemWithTTL(
        final String tableName, final String primaryKeyValue, final String
 sortKeyValue) {
        // 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 + (DAYS_TO_EXPIRE * SECONDS_PER_DAY);

        // Create the key map for the item to update
        final Map<String, AttributeValue> keyMap = new HashMap<>();
        keyMap.put(PRIMARY_KEY_ATTR,
 AttributeValue.builder().s(primaryKeyValue).build());
        keyMap.put(SORT_KEY_ATTR, AttributeValue.builder().s(sortKeyValue).build());

        // Create the expression attribute values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            ":c", AttributeValue.builder().n(String.valueOf(currentTime)).build());
        expressionAttributeValues.put(
            ":e", AttributeValue.builder().n(String.valueOf(expireDate)).build());

        final UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(keyMap)
            .updateExpression(UPDATE_EXPRESSION)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final UpdateItemResponse response = dynamoDbClient.updateItem(request);
            System.out.println(String.format(SUCCESS_MESSAGE, tableName));
```

```
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateItem을 참조하세요.

API Gateway를 사용하여 Lambda 함수 간접 호출

다음 코드 예제에서는 Amazon API Gateway에서 호출한 AWS Lambda 함수를 생성하는 방법을 보여줍니다.

SDK for Java 2.x

Lambda Java 런타임 API를 사용하여 AWS Lambda 함수를 생성하는 방법을 보여줍니다. 이 예제에서는 다양한 AWS 서비스를 호출하여 특정 사용 사례를 수행합니다. 이 예제에서는 Amazon API Gateway에서 간접 호출한 Lambda 함수를 생성하여 작업 기념일에 대한 Amazon DynamoDB 테이블을 스캔하고 Amazon Simple Notification Service(Amazon SNS)를 사용하여 직원에게 1주년 기념일을 축하하는 문자 메시지를 전송하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Step Functions를 사용하여 Lambda 함수 간접 호출

다음 코드 예제에서는 AWS Lambda 함수를 순차적으로 호출하는 AWS Step Functions 상태 시스템을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

AWS Step Functions 및를 사용하여 AWS 서버리스 워크플로를 생성하는 방법을 보여줍니다 AWS SDK for Java 2.x. 각 워크플로 단계는 AWS Lambda 함수를 사용하여 구현됩니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- DynamoDB
- Lambda
- Amazon SES
- 단계 함수

원자성 카운터 작업 사용

다음 코드 예제에서는 DynamoDB에서 원자성 카운터 작업을 사용하는 방법을 보여줍니다.

- ADD 및 SET 작업을 사용하여 원자적으로 카운터를 증가시킵니다.
- 존재하지 않을 수 있는 카운터를 안전하게 증가시킵니다.
- 카운터 작업에 대한 낙관적 잠금을 구현합니다.

SDK for Java 2.x

를 사용하여 원자성 카운터 작업을 시연합니다 AWS SDK for Java 2.x.

```
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.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ReturnValue;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.Map;

    /**
     * Increments a counter using the ADD operation.
     *
```

```
     * <p>This method demonstrates how to use the ADD operation to atomically
     * increment a counter attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param counterName The name of the counter attribute
     * @param incrementValue The value to increment by
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse incrementCounterWithAdd(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String counterName,
        int incrementValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("ADD #counterName :increment")
            .expressionAttributeNames(Map.of("#counterName", counterName))
            .expressionAttributeValues(Map.of(
                ":increment",
                AttributeValue.builder().n(String.valueOf(incrementValue)).build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Increments a counter using the SET operation.
     *
     * <p>This method demonstrates how to use the SET operation with an expression
     * to increment a counter attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param counterName The name of the counter attribute
```

```
     * @param incrementValue The value to increment by
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse incrementCounterWithSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String counterName,
        int incrementValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #counterName = #counterName + :increment")
            .expressionAttributeNames(Map.of("#counterName", counterName))
            .expressionAttributeValues(Map.of(
                ":increment",
                AttributeValue.builder().n(String.valueOf(incrementValue)).build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Increments a counter safely, handling the case where the counter doesn't
exist yet.
     *
     * <p>This method demonstrates how to use if_not_exists to safely increment a
counter
     * that may not exist yet.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param counterName The name of the counter attribute
     * @param incrementValue The value to increment by
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse incrementCounterSafely(
```

```
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String counterName,
        int incrementValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #counterName = if_not_exists(#counterName, :zero)
+ :increment")
            .expressionAttributeNames(Map.of("#counterName", counterName))
            .expressionAttributeValues(Map.of(
                ":increment",

AttributeValue.builder().n(String.valueOf(incrementValue)).build(),
                ":zero", AttributeValue.builder().n("0").build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Decrements a counter safely, ensuring it doesn't go below zero.
     *
     * <p>This method demonstrates how to use a condition expression to safely
     * decrement a counter without going below zero.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param counterName The name of the counter attribute
     * @param decrementValue The value to decrement by
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation or if the
counter would go below zero
     */
    public static UpdateItemResponse decrementCounterSafely(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
```

```
        String counterName,
        int decrementValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #counterName = #counterName - :decrement")
            .conditionExpression("#counterName >= :decrement")
            .expressionAttributeNames(Map.of("#counterName", counterName))
            .expressionAttributeValues(Map.of(
                ":decrement",
                AttributeValue.builder().n(String.valueOf(decrementValue)).build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Compares the ADD and SET approaches for incrementing counters.
     *
     * <p>This method demonstrates the differences between using ADD and SET
     * for incrementing counters in DynamoDB.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @return Map containing the comparison results
     */
    public static Map<String, Object> compareAddVsSet(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key) {

        Map<String, Object> results = new HashMap<>();

        try {
            // Reset counters to ensure a fair comparison
            UpdateItemRequest resetRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET AddCounter = :zero, SetCounter = :zero")
                .expressionAttributeValues(
```

```
                    Map.of(":zero", AttributeValue.builder().n("0").build()))
                .build();

            dynamoDbClient.updateItem(resetRequest);

            // Increment with ADD
            long addStartTime = System.nanoTime();
            UpdateItemResponse addResponse = incrementCounterWithAdd(dynamoDbClient,
tableName, key, "AddCounter", 1);
            long addEndTime = System.nanoTime();
            long addDuration = addEndTime - addStartTime;

            // Increment with SET
            long setStartTime = System.nanoTime();
            UpdateItemResponse setResponse = incrementCounterWithSet(dynamoDbClient,
tableName, key, "SetCounter", 1);
            long setEndTime = System.nanoTime();
            long setDuration = setEndTime - setStartTime;

            // Record results
            results.put("addResponse", addResponse);
            results.put("setResponse", setResponse);
            results.put("addDuration", addDuration);
            results.put("setDuration", setDuration);
            results.put("success", true);

        } catch (DynamoDbException e) {
            results.put("success", false);
            results.put("error", e.getMessage());
        }

        return results;
    }

    /**
     * Gets the current value of a counter attribute.
     *
     * <p>Helper method to retrieve the current value of a counter attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @param counterName The name of the counter attribute
     * @return The counter value or null if not found
```

```
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Integer getCounterValue(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, String counterName) {

        // Define the get parameters
        GetItemRequest request = GetItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .projectionExpression(counterName)
            .build();

        // Perform the get operation
        GetItemResponse response = dynamoDbClient.getItem(request);

        // Return the counter value if it exists, otherwise null
        if (response.item() != null && response.item().containsKey(counterName)) {
            return Integer.parseInt(response.item().get(counterName).n());
        }

        return null;
    }
```

를 사용한 원자성 카운터 작업 사용의 예입니다 AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating atomic counter operations in DynamoDB");

        try {
            // Example 1: Increment a counter using ADD
            System.out.println("\nExample 1: Incrementing a counter using ADD");
            UpdateItemResponse addResponse = incrementCounterWithAdd(dynamoDbClient,
tableName, key, "ViewCount", 1);

            System.out.println("Updated counter: " + addResponse.attributes());
```

```
            // Example 2: Increment a counter using SET
            System.out.println("\nExample 2: Incrementing a counter using SET");
            UpdateItemResponse setResponse = incrementCounterWithSet(dynamoDbClient,
tableName, key, "LikeCount", 1);

            System.out.println("Updated counter: " + setResponse.attributes());

            // Example 3: Increment a counter safely
            System.out.println("\nExample 3: Incrementing a counter safely");
            UpdateItemResponse safeResponse = incrementCounterSafely(dynamoDbClient,
tableName, key, "ShareCount", 1);

            System.out.println("Updated counter: " + safeResponse.attributes());

            // Example 4: Decrement a counter safely
            System.out.println("\nExample 4: Decrementing a counter safely");
            try {
                UpdateItemResponse decrementResponse =
                    decrementCounterSafely(dynamoDbClient, tableName, key,
"InventoryCount", 1);

                System.out.println("Updated counter: " +
decrementResponse.attributes());
            } catch (DynamoDbException e) {
                if (e.getMessage().contains("ConditionalCheckFailed")) {
                    System.out.println("Cannot decrement counter below zero");
                } else {
                    throw e;
                }
            }

            // Example 5: Compare ADD vs SET
            System.out.println("\nExample 5: Comparing ADD vs SET");
            Map<String, Object> comparison = compareAddVsSet(dynamoDbClient,
tableName, key);

            if ((boolean) comparison.get("success")) {
                System.out.println("ADD duration: " + comparison.get("addDuration")
+ " ns");
                System.out.println("SET duration: " + comparison.get("setDuration")
+ " ns");
                System.out.println("ADD response: " +
comparison.get("addResponse"));
```

```
                System.out.println("SET response: " +
comparison.get("setResponse"));
            } else {
                System.out.println("Comparison failed: " + comparison.get("error"));
            }

            // Explain atomic counter operations
            System.out.println("\nKey points about DynamoDB atomic counter
operations:");
            System.out.println("1. Both ADD and SET can be used for atomic
counters");
            System.out.println("2. ADD is more concise for simple increments");
            System.out.println("3. SET with an expression is more flexible for
complex operations");
            System.out.println("4. Use if_not_exists to handle the case where the
counter doesn't exist yet");
            System.out.println("5. Use condition expressions to prevent counters
from going below zero");
            System.out.println("6. Atomic operations are guaranteed to be isolated
from other writes");
            System.out.println("7. ADD can only be used with number and set data
types");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateItem을 참조하세요.

조건부 작업 사용

다음 코드 예제에서는 DynamoDB에서 조건부 작업을 사용하는 방법을 보여줍니다.

- 데이터 덮어쓰기를 방지하기 위해 조건부 쓰기를 구현합니다.
- 조건 표현식을 사용하여 비즈니스 규칙을 적용합니다.
- 조건부 검사 실패를 원활하게 처리합니다.

SDK for Java 2.x

를 사용하여 조건부 작업을 시연합니다 AWS SDK for Java 2.x.

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import
 software.amazon.awssdk.services.dynamodb.model.ConditionalCheckFailedException;
import software.amazon.awssdk.services.dynamodb.model.DeleteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.DeleteItemResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ReturnValue;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.Map;

    /**
     * Performs a conditional update on an item.
     *
     * <p>This method demonstrates how to use a condition expression to update an
 item
     * only if a specific condition is met.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param conditionAttribute The attribute to check in the condition
     * @param conditionValue The value to compare against
     * @param updateAttribute The attribute to update
     * @param updateValue The new value to set
     * @return Map containing the operation result and status
     */
    public static Map<String, Object> conditionalUpdate(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String conditionAttribute,
        AttributeValue conditionValue,
        String updateAttribute,
        AttributeValue updateValue) {
```

```
        Map<String, Object> result = new HashMap<>();

        try {
            // Define the update parameters
            UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #updateAttr = :updateVal")
                .conditionExpression("#condAttr = :condVal")
                .expressionAttributeNames(Map.of(
                    "#condAttr", conditionAttribute,
                    "#updateAttr", updateAttribute))
                .expressionAttributeValues(Map.of(
                    ":condVal", conditionValue,
                    ":updateVal", updateValue))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            // Perform the update operation
            UpdateItemResponse response = dynamoDbClient.updateItem(request);

            // Record success result
            result.put("success", true);
            result.put("message", "Condition was met and update was performed");
            result.put("attributes", response.attributes());

        } catch (ConditionalCheckFailedException e) {
            // Record failure due to condition not being met
            result.put("success", false);
            result.put("message", "Condition was not met, update was not
performed");
            result.put("error", "ConditionalCheckFailedException");

        } catch (DynamoDbException e) {
            // Record failure due to other errors
            result.put("success", false);
            result.put("message", "Error occurred: " + e.getMessage());
            result.put("error", e.getClass().getSimpleName());
        }

        return result;
    }
```

```
    /**
     * Performs a conditional delete on an item.
     *
     * <p>This method demonstrates how to use a condition expression to delete an
item
     * only if a specific condition is met.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to delete
     * @param conditionAttribute The attribute to check in the condition
     * @param conditionValue The value to compare against
     * @return Map containing the operation result and status
     */
    public static Map<String, Object> conditionalDelete(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String conditionAttribute,
        AttributeValue conditionValue) {

        Map<String, Object> result = new HashMap<>();

        try {
            // Define the delete parameters
            DeleteItemRequest request = DeleteItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .conditionExpression("#condAttr = :condVal")
                .expressionAttributeNames(Map.of("#condAttr", conditionAttribute))
                .expressionAttributeValues(Map.of(":condVal", conditionValue))
                .returnValues(ReturnValue.ALL_OLD)
                .build();

            // Perform the delete operation
            DeleteItemResponse response = dynamoDbClient.deleteItem(request);

            // Record success result
            result.put("success", true);
            result.put("message", "Condition was met and delete was performed");
            result.put("attributes", response.attributes());

        } catch (ConditionalCheckFailedException e) {
            // Record failure due to condition not being met
```

```
            result.put("success", false);
            result.put("message", "Condition was not met, delete was not
performed");
            result.put("error", "ConditionalCheckFailedException");

        } catch (DynamoDbException e) {
            // Record failure due to other errors
            result.put("success", false);
            result.put("message", "Error occurred: " + e.getMessage());
            result.put("error", e.getClass().getSimpleName());
        }

        return result;
    }

    /**
     * Demonstrates optimistic locking using a version attribute.
     *
     * <p>This method shows how to implement optimistic locking by using a version
     * attribute that is incremented with each update.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param versionAttribute The name of the version attribute
     * @return Map containing the operation result
     */
    public static Map<String, Object> optimisticLockingExample(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, String versionAttribute) {

        Map<String, Object> result = new HashMap<>();

        try {
            // Get the current version of the item
            GetItemRequest getRequest = GetItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .projectionExpression(versionAttribute)
                .build();

            GetItemResponse getResponse = dynamoDbClient.getItem(getRequest);

            // Check if the item exists
```

```
            if (getResponse.item() == null || !
getResponse.item().containsKey(versionAttribute)) {
                // Item doesn't exist or doesn't have a version attribute
                // Initialize with version 1
                UpdateItemRequest initRequest = UpdateItemRequest.builder()
                    .tableName(tableName)
                    .key(key)
                    .updateExpression("SET #verAttr = :newVer, #dataAttr = :data")
                    .expressionAttributeNames(Map.of("#verAttr", versionAttribute,
 "#dataAttr", "Data"))
                    .expressionAttributeValues(Map.of(
                        ":newVer", AttributeValue.builder().n("1").build(),
                        ":data", AttributeValue.builder().s("Initial
 data").build()))
                    .returnValues(ReturnValue.UPDATED_NEW)
                    .build();

                UpdateItemResponse initResponse =
 dynamoDbClient.updateItem(initRequest);

                result.put("operation", "initialize");
                result.put("success", true);
                result.put("attributes", initResponse.attributes());

                return result;
            }

            // Get the current version number
            int currentVersion =
                Integer.parseInt(getResponse.item().get(versionAttribute).n());
            int newVersion = currentVersion + 1;

            // Update the item with a condition on the version
            UpdateItemRequest updateRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #verAttr = :newVer, #dataAttr = :newData")
                .conditionExpression("#verAttr = :curVer")
                .expressionAttributeNames(Map.of("#verAttr", versionAttribute,
 "#dataAttr", "Data"))
                .expressionAttributeValues(Map.of(
                    ":curVer",
                        AttributeValue.builder()
                            .n(String.valueOf(currentVersion))
```

```
                            .build(),
                    ":newVer",

AttributeValue.builder().n(String.valueOf(newVersion)).build(),
                    ":newData",
                        AttributeValue.builder()
                            .s("Updated data at version " + newVersion)
                            .build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse updateResponse =
dynamoDbClient.updateItem(updateRequest);

            // Record success result
            result.put("operation", "update");
            result.put("success", true);
            result.put("oldVersion", currentVersion);
            result.put("newVersion", newVersion);
            result.put("attributes", updateResponse.attributes());

        } catch (ConditionalCheckFailedException e) {
            // Record failure due to version mismatch
            result.put("operation", "update");
            result.put("success", false);
            result.put("message", "Version mismatch, another process may have
updated the item");
            result.put("error", "ConditionalCheckFailedException");

        } catch (DynamoDbException e) {
            // Record failure due to other errors
            result.put("operation", "update");
            result.put("success", false);
            result.put("message", "Error occurred: " + e.getMessage());
            result.put("error", e.getClass().getSimpleName());
        }

        return result;
    }

    /**
     * Performs a conditional update with multiple conditions.
     *
```

```
    * <p>This method demonstrates how to use multiple conditions in a condition
expression.
    *
    * @param dynamoDbClient The DynamoDB client
    * @param tableName The name of the DynamoDB table
    * @param key The key of the item to update
    * @param conditions Map of attribute names to values for conditions
    * @param updateAttribute The attribute to update
    * @param updateValue The new value to set
    * @return Map containing the operation result and status
    */
   public static Map<String, Object> conditionalUpdateWithMultipleConditions(
       DynamoDbClient dynamoDbClient,
       String tableName,
       Map<String, AttributeValue> key,
       Map<String, AttributeValue> conditions,
       String updateAttribute,
       AttributeValue updateValue) {

       Map<String, Object> result = new HashMap<>();

       try {
           // Build the condition expression and attribute names/values
           StringBuilder conditionExpression = new StringBuilder();
           Map<String, String> expressionAttributeNames = new HashMap<>();
           Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();

           // Add update attribute
           expressionAttributeNames.put("#updateAttr", updateAttribute);
           expressionAttributeValues.put(":updateVal", updateValue);

           // Add conditions
           int i = 0;
           for (Map.Entry<String, AttributeValue> condition :
conditions.entrySet()) {
               String attrName = condition.getKey();
               AttributeValue attrValue = condition.getValue();

               String nameKey = "#cond" + i;
               String valueKey = ":val" + i;

               expressionAttributeNames.put(nameKey, attrName);
               expressionAttributeValues.put(valueKey, attrValue);
```

```
                // Add AND between conditions (except for the first one)
                if (i > 0) {
                    conditionExpression.append(" AND ");
                }

                conditionExpression.append(nameKey).append(" = ").append(valueKey);
                i++;
            }

            // Define the update parameters
            UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #updateAttr = :updateVal")
                .conditionExpression(conditionExpression.toString())
                .expressionAttributeNames(expressionAttributeNames)
                .expressionAttributeValues(expressionAttributeValues)
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            // Perform the update operation
            UpdateItemResponse response = dynamoDbClient.updateItem(request);

            // Record success result
            result.put("success", true);
            result.put("message", "All conditions were met and update was
performed");
            result.put("attributes", response.attributes());

        } catch (ConditionalCheckFailedException e) {
            // Record failure due to condition not being met
            result.put("success", false);
            result.put("message", "One or more conditions were not met, update was
not performed");
            result.put("error", "ConditionalCheckFailedException");

        } catch (DynamoDbException e) {
            // Record failure due to other errors
            result.put("success", false);
            result.put("message", "Error occurred: " + e.getMessage());
            result.put("error", e.getClass().getSimpleName());
        }

        return result;
```

```
    }
```

를 사용한 조건부 작업 사용의 예입니다 AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating conditional operations in DynamoDB");

        try {
            // Example 1: Conditional update
            System.out.println("\nExample 1: Conditional update");
            Map<String, Object> updateResult = conditionalUpdate(
                dynamoDbClient,
                tableName,
                key,
                "InStock",
                AttributeValue.builder().bool(true).build(),
                "Status",
                AttributeValue.builder().s("Available").build());

            System.out.println("Update result: " + updateResult.get("message"));
            if ((boolean) updateResult.get("success")) {
                System.out.println("Updated attributes: " +
updateResult.get("attributes"));
            }

            // Example 2: Conditional delete
            System.out.println("\nExample 2: Conditional delete");
            Map<String, Object> deleteResult = conditionalDelete(
                dynamoDbClient,
                tableName,
                key,
                "Status",
                AttributeValue.builder().s("Discontinued").build());

            System.out.println("Delete result: " + deleteResult.get("message"));
            if ((boolean) deleteResult.get("success")) {
```

```
                System.out.println("Deleted item: " +
deleteResult.get("attributes"));
            }

            // Example 3: Optimistic locking
            System.out.println("\nExample 3: Optimistic locking");
            Map<String, Object> lockingResult =
optimisticLockingExample(dynamoDbClient, tableName, key, "Version");

            System.out.println("Optimistic locking result:");
            System.out.println("  Operation: " + lockingResult.get("operation"));
            System.out.println("  Success: " + lockingResult.get("success"));
            if (lockingResult.get("operation").equals("update") && (boolean)
lockingResult.get("success")) {
                System.out.println("  Old version: " +
lockingResult.get("oldVersion"));
                System.out.println("  New version: " +
lockingResult.get("newVersion"));
            }
            System.out.println("  Attributes: " + lockingResult.get("attributes"));

            // Example 4: Multiple conditions
            System.out.println("\nExample 4: Multiple conditions");
            Map<String, AttributeValue> conditions = new HashMap<>();
            conditions.put("Price", AttributeValue.builder().n("199.99").build());
            conditions.put("Category",
AttributeValue.builder().s("Electronics").build());

            Map<String, Object> multiConditionResult =
conditionalUpdateWithMultipleConditions(
                dynamoDbClient,
                tableName,
                key,
                conditions,
                "OnSale",
                AttributeValue.builder().bool(true).build());

            System.out.println("Multiple conditions result: " +
multiConditionResult.get("message"));
            if ((boolean) multiConditionResult.get("success")) {
                System.out.println("Updated attributes: " +
multiConditionResult.get("attributes"));
            }
```

```
            // Explain conditional operations
            System.out.println("\nKey points about DynamoDB conditional
operations:");
            System.out.println("1. Conditional operations only succeed if the
condition is met");
            System.out.println("2. ConditionalCheckFailedException is thrown when
the condition fails");
            System.out.println("3. No changes are made to the item if the condition
fails");
            System.out.println("4. Conditions can be used with update, delete, and
put operations");
            System.out.println("5. Multiple conditions can be combined with AND and
OR");
            System.out.println("6. Optimistic locking can be implemented using a
version attribute");
            System.out.println(
                "7. Conditional operations consume the same amount of write capacity
whether they succeed or fail");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - DeleteItem
  - PutItem
  - UpdateItem

표현식 속성 이름 사용

다음 코드 예제에서는 DynamoDB에서 표현식 속성 이름을 사용하는 방법을 보여줍니다.

- DynamoDB 표현식에서 예약어로 작업합니다.
- 표현식 속성 이름 자리 표시자를 사용합니다.
- 속성 이름의 특수 문자를 처리합니다.

SDK for Java 2.x

를 사용하여 표현식 속성 이름을 보여줍니다 AWS SDK for Java 2.x.

```
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.ScanRequest;
import software.amazon.awssdk.services.dynamodb.model.ScanResponse;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    /**
     * Updates an attribute that is a reserved word in DynamoDB.
     *
     * <p>This method demonstrates how to use expression attribute names to update
     * attributes that are reserved words in DynamoDB.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param reservedWordAttribute The reserved word attribute to update
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateReservedWordAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String reservedWordAttribute,
        AttributeValue value) {

        // Define the update parameters using expression attribute names
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
```

```
            .updateExpression("SET #attr = :value")
            .expressionAttributeNames(Map.of("#attr", reservedWordAttribute))
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Updates an attribute that contains special characters.
     *
     * <p>This method demonstrates how to use expression attribute names to update
     * attributes that contain special characters.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param specialCharAttribute The attribute with special characters to update
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateSpecialCharacterAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String specialCharAttribute,
        AttributeValue value) {

        // Define the update parameters using expression attribute names
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attr = :value")
            .expressionAttributeNames(Map.of("#attr", specialCharAttribute))
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }
```

```
/**
 * Queries items using an attribute that is a reserved word.
 *
 * <p>This method demonstrates how to use expression attribute names in a query
 * when the attribute is a reserved word.
 *
 * @param dynamoDbClient The DynamoDB client
 * @param tableName The name of the DynamoDB table
 * @param partitionKeyName The name of the partition key attribute
 * @param partitionKeyValue The value of the partition key
 * @param reservedWordAttribute The reserved word attribute to filter on
 * @param value The value to compare against
 * @return The response from DynamoDB
 * @throws DynamoDbException if an error occurs during the operation
 */
public static QueryResponse queryWithReservedWordAttribute(
    DynamoDbClient dynamoDbClient,
    String tableName,
    String partitionKeyName,
    AttributeValue partitionKeyValue,
    String reservedWordAttribute,
    AttributeValue value) {

    // Define the query parameters using expression attribute names
    Map<String, String> expressionAttributeNames = new HashMap<>();
    expressionAttributeNames.put("#pkName", partitionKeyName);
    expressionAttributeNames.put("#attr", reservedWordAttribute);

    Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
    expressionAttributeValues.put(":pkValue", partitionKeyValue);
    expressionAttributeValues.put(":value", value);

    QueryRequest request = QueryRequest.builder()
        .tableName(tableName)
        .keyConditionExpression("#pkName = :pkValue")
        .filterExpression("#attr = :value")
        .expressionAttributeNames(expressionAttributeNames)
        .expressionAttributeValues(expressionAttributeValues)
        .build();

    // Perform the query operation
    return dynamoDbClient.query(request);
}
```

```
    /**
     * Updates a nested attribute with a path that contains reserved words.
     *
     * <p>This method demonstrates how to use expression attribute names to update
     * nested attributes where the path contains reserved words.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param attributePath The path to the nested attribute as an array
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateNestedReservedWordAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        List<String> attributePath,
        AttributeValue value) {

        // Create expression attribute names for each part of the path
        Map<String, String> expressionAttributeNames = new HashMap<>();
        for (int i = 0; i < attributePath.size(); i++) {
            expressionAttributeNames.put("#attr" + i, attributePath.get(i));
        }

        // Build the attribute path using the expression attribute names
        StringBuilder attributePathExpression = new StringBuilder();
        for (int i = 0; i < attributePath.size(); i++) {
            if (i > 0) {
                attributePathExpression.append(".");
            }
            attributePathExpression.append("#attr").append(i);
        }

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET " + attributePathExpression.toString() + "
= :value")
            .expressionAttributeNames(expressionAttributeNames)
```

```
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Scans a table with multiple attribute name placeholders.
     *
     * <p>This method demonstrates how to use multiple expression attribute names
     * in a complex filter expression.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param filters Object mapping attribute names to filter values
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static ScanResponse scanWithMultipleAttributeNames(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
filters) {

        // Create expression attribute names and values
        Map<String, String> expressionAttributeNames = new HashMap<>();
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
        StringBuilder filterExpression = new StringBuilder();

        // Build the filter expression
        int index = 0;
        for (Map.Entry<String, AttributeValue> entry : filters.entrySet()) {
            String attrName = entry.getKey();
            AttributeValue attrValue = entry.getValue();

            String nameKey = "#attr" + index;
            String valueKey = ":val" + index;

            expressionAttributeNames.put(nameKey, attrName);
            expressionAttributeValues.put(valueKey, attrValue);

            // Add AND between conditions (except for the first one)
            if (index > 0) {
                filterExpression.append(" AND ");
```

```
            }

            filterExpression.append(nameKey).append(" = ").append(valueKey);
            index++;
        }

        // Define the scan parameters
        ScanRequest request = ScanRequest.builder()
            .tableName(tableName)
            .filterExpression(filterExpression.toString())
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the scan operation
        return dynamoDbClient.scan(request);
    }
```

에서 표현식 속성 이름의 사용 예제입니다 AWS SDK for Java 2.x.

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating expression attribute names in DynamoDB");

        try {
            // Example 1: Update an attribute that is a reserved word
            System.out.println("\nExample 1: Updating an attribute that is a
reserved word");
            UpdateItemResponse response1 = updateReservedWordAttribute(
                dynamoDbClient,
                tableName,
                key,
                "Size", // "SIZE" is a reserved word in DynamoDB
                AttributeValue.builder().s("Large").build());

            System.out.println("Updated attribute: " + response1.attributes());

            // Example 2: Update an attribute with special characters
```

```
            System.out.println("\nExample 2: Updating an attribute with special
characters");
            UpdateItemResponse response2 = updateSpecialCharacterAttribute(
                dynamoDbClient,
                tableName,
                key,
                "Product-Type", // Contains a hyphen, which is a special character
                AttributeValue.builder().s("Electronics").build());

            System.out.println("Updated attribute: " + response2.attributes());

            // Example 3: Query with a reserved word attribute
            System.out.println("\nExample 3: Querying with a reserved word
attribute");
            QueryResponse response3 = queryWithReservedWordAttribute(
                dynamoDbClient,
                tableName,
                "Category",
                AttributeValue.builder().s("Electronics").build(),
                "Count", // "COUNT" is a reserved word in DynamoDB
                AttributeValue.builder().n("10").build());

            System.out.println("Found " + response3.count() + " items");

            // Example 4: Update a nested attribute with reserved words in the path
            System.out.println("\nExample 4: Updating a nested attribute with
reserved words in the path");
            UpdateItemResponse response4 = updateNestedReservedWordAttribute(
                dynamoDbClient,
                tableName,
                key,
                Arrays.asList("Dimensions", "Size", "Height"), // "SIZE" is a
reserved word
                AttributeValue.builder().n("30").build());

            System.out.println("Updated nested attribute: " +
response4.attributes());

            // Example 5: Scan with multiple attribute name placeholders
            System.out.println("\nExample 5: Scanning with multiple attribute name
placeholders");
            Map<String, AttributeValue> filters = new HashMap<>();
            filters.put("Size", AttributeValue.builder().s("Large").build());
            filters.put("Count", AttributeValue.builder().n("10").build());
```

```
            filters.put(
                "Product-Type", AttributeValue.builder().s("Electronics").build());

            ScanResponse response5 = scanWithMultipleAttributeNames(dynamoDbClient,
tableName, filters);

            System.out.println("Found " + response5.count() + " items");

            // Show some common reserved words
            System.out.println("\nSome common DynamoDB reserved words:");
            List<String> commonReservedWords = getDynamoDBReservedWords();
            System.out.println(String.join(", ", commonReservedWords));

            // Explain expression attribute names
            System.out.println("\nKey points about expression attribute names:");
            System.out.println("1. Use expression attribute names (#name) for
reserved words");
            System.out.println("2. Use expression attribute names for attributes
with special characters");
            System.out.println(
                "3. Special characters include: spaces, hyphens, dots, and other
non-alphanumeric characters");
            System.out.println("4. Expression attribute names are required for
nested attributes with reserved words");
            System.out.println("5. You can use multiple expression attribute names
in a single expression");
            System.out.println("6. Expression attribute names are case-sensitive");
            System.out.println("7. Expression attribute names are only used in
expressions, not in the actual data");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - Query
  - UpdateItem

예약된 이벤트를 사용하여 Lambda 함수 간접 호출

다음 코드 예제에서는 Amazon EventBridge 예약 이벤트에서 호출된 AWS Lambda 함수를 생성하는 방법을 보여줍니다.

SDK for Java 2.x

AWS Lambda 함수를 호출하는 Amazon EventBridge 예약 이벤트를 생성하는 방법을 보여줍니다. Lambda 함수가 간접 호출될 때 cron 표현식을 사용하여 일정을 예약하도록 EventBridge를 구성합니다. 이 예제에서는 Lambda Java 런타임 API를 사용하여 Lambda 함수를 생성합니다. 이 예제에서는 다양한 AWS 서비스를 호출하여 특정 사용 사례를 수행합니다. 이 예제에서는 1주년 기념일에 직원에게 축하하는 모바일 문자 메시지를 전송하는 앱을 생성하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

서버리스 예제

DynamoDB 트리거에서 간접적으로 Lambda 함수 간접 호출

다음 코드 예제에서는 DynamoDB 스트림에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 DynamoDB 페이로드를 검색하고 레코드 콘텐츠를 로깅합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Java를 사용하여 Lambda로 DynamoDB 이벤트 소비

```
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()));
    }
}
```

DynamoDB 트리거로 Lambda 함수에 대한 배치 항목 실패 보고

다음 코드 예제에서는 DynamoDB 스트림에서 이벤트를 수신하는 Lambda 함수에 대한 부분 배치 응답을 구현하는 방법을 보여줍니다. 이 함수는 응답으로 배치 항목 실패를 보고하고 나중에 해당 메시지를 다시 시도하도록 Lambda에 신호를 보냅니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Java를 사용하여 Lambda로 DynamoDB 배치 항목 실패 보고.

```
// 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.util.ArrayList;
import java.util.List;

public class ProcessDynamodbRecords implements RequestHandler<DynamodbEvent,
 StreamsEventResponse> {

    @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();
    }
}
```

AWS 커뮤니티 기여

서버리스 애플리케이션 빌드 및 테스트

다음 코드 예제에서는 Lambda 및 DynamoDB와 함께 API 게이트웨이를 사용하여 서버리스 애플리케이션을 빌드하고 테스트하는 방법을 보여줍니다.

SDK for Java 2.x

Java SDK를 사용하여 Lambda 및 DynamoDB가 포함된 API 게이트웨이로 구성된 서버리스 애플리케이션을 빌드하고 테스트하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda

Java 2.x용 SDK를 사용하는 Amazon EC2 예제

다음 코드 예제에서는 Amazon EC2와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시작하기

시작하기

Hello Amazon EC2

다음 코드 예제에서는 Amazon EC2 사용을 시작하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Asynchronously describes the security groups for the specified group ID.
    *
    * @param groupName the name of the security group to describe
    * @return a {@link CompletableFuture} that represents the asynchronous
operation
    *         of describing the security groups. The future will complete with a
    *         {@link DescribeSecurityGroupsResponse} object that contains the
    *         security group information.
    */
   public CompletableFuture<String> describeSecurityGroupArnByNameAsync(String
groupName) {
       DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
           .groupNames(groupName)
           .build();

       DescribeSecurityGroupsPublisher paginator =
getAsyncClient().describeSecurityGroupsPaginator(request);
       AtomicReference<String> groupIdRef = new AtomicReference<>();
       return paginator.subscribe(response -> {
           response.securityGroups().stream()
```

```
                .filter(securityGroup ->
securityGroup.groupName().equals(groupName))
                .findFirst()
                .ifPresent(securityGroup ->
groupIdRef.set(securityGroup.groupId()));
        }).thenApply(v -> {
            String groupId = groupIdRef.get();
            if (groupId == null) {
                throw new RuntimeException("No security group found with the name: "
+ groupName);
            }
            return groupId;
        }).exceptionally(ex -> {
            logger.info("Failed to describe security group: " + ex.getMessage());
            throw new RuntimeException("Failed to describe security group", ex);
        });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeSecurityGroups를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 키 페어 및 보안 그룹을 생성합니다.
- Amazon Machine Image(AMI) 및 호환되는 인스턴스 유형을 선택한 다음 인스턴스를 생성합니다.
- 인스턴스를 중지한 후 다시 시작합니다.
- 인스턴스와 탄력적 IP 주소 연결.
- SSH로 인스턴스에 연결한 다음 리소스를 정리합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 시나리오를 실행합니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.ec2.model.CreateKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.DeleteKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.DescribeKeyPairsResponse;
import software.amazon.awssdk.services.ec2.model.DisassociateAddressResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.ReleaseAddressResponse;
import software.amazon.awssdk.services.ssm.model.GetParametersByPathResponse;
import software.amazon.awssdk.services.ssm.model.Parameter;

import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including 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 additional 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", "-");
    private static final Logger logger = LoggerFactory.getLogger(EC2Scenario.class);
    public static void main(String[] args) throws InterruptedException,
 UnknownHostException {

        logger.info("""
            Usage:
               <keyName> <fileName> <groupName> <groupDesc>

            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
            """);

        Scanner scanner = new Scanner(System.in);
        EC2Actions ec2Actions = new EC2Actions();

        String keyName = "TestKeyPair7" ;
        String fileName = "ec2Key.pem";
        String groupName = "TestSecGroup7" ;
        String groupDesc = "Test Group" ;
        String vpcId = ec2Actions.describeFirstEC2VpcAsync().join().vpcId();
        InetAddress localAddress = InetAddress.getLocalHost();
        String myIpAddress = localAddress.getHostAddress();

        logger.info("""
            Amazon Elastic Compute Cloud (EC2) is a web service that provides
 secure, resizable compute
            capacity in the cloud. It allows developers and organizations to easily
 launch and manage
            virtual server instances, known as EC2 instances, to run their
 applications.
```

```
            EC2 provides a wide range of instance types, each with different
 compute, memory,
            and storage capabilities, to meet the diverse needs of various
 workloads. Developers
            can choose the appropriate instance type based on their application's
 requirements,
            such as high-performance computing, memory-intensive tasks, or GPU-
accelerated workloads.

            The `Ec2AsyncClient` interface in the AWS SDK for Java 2.x provides a
 set of methods to
            programmatically interact with the Amazon EC2 service. This allows
 developers to
            automate the provisioning, management, and monitoring of EC2 instances
 as part of their
            application deployment pipelines. With EC2, teams can focus on building
 and deploying
            their applications without having to worry about the underlying
 infrastructure
            required to host and manage physical servers.

            This scenario walks you through how to perform key operations for this
 service.
            Let's get started...
            """);

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("1. Create an RSA key pair and save the private key material as
 a .pem file.");
        logger.info("""
            An RSA key pair for Amazon EC2 is a security mechanism used to
 authenticate and secure
            access to your EC2 instances. It consists of a public key and a private
 key,
            which are generated as a pair.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<CreateKeyPairResponse> future =
 ec2Actions.createKeyPairAsync(keyName, fileName);
            CreateKeyPairResponse response = future.join();
```

```
            logger.info("Key Pair successfully created. Key Fingerprint: " +
response.keyFingerprint());

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                if (ec2Ex.getMessage().contains("already exists")) {
                    // Key pair already exists.
                    logger.info("The key pair '" + keyName + "' already exists.
Moving on...");
                } else {
                    logger.info("EC2 error occurred: Error message: {}, Error code
{}", ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                    return;
                }
            } else {
                logger.info("An unexpected error occurred: " + (rt.getMessage()));
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. List key pairs.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<DescribeKeyPairsResponse> future =
ec2Actions.describeKeysAsync();
            DescribeKeyPairsResponse keyPairsResponse = future.join();
            keyPairsResponse.keyPairs().forEach(keyPair -> logger.info(
                "Found key pair with name {} and fingerprint {}",
                keyPair.keyName(),
                keyPair.keyFingerprint()));

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Error message: {}, Error code {}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", (cause != null ?
cause.getMessage() : rt.getMessage()));
```

```
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Create a security group.");
        logger.info("""
            An AWS EC2 Security Group is a virtual firewall that controls the
            inbound and outbound traffic to an EC2 instance. It acts as a first
line
            of defense for your EC2 instances, allowing you to specify the rules
that
            govern the network traffic entering and leaving your instances.
           """);
        waitForInputToContinue(scanner);
        String groupId = "";
        try {
            CompletableFuture<String> future =
ec2Actions.createSecurityGroupAsync(groupName, groupDesc, vpcId, myIpAddress);
            future.join();
            logger.info("Created security group") ;

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                if (ec2Ex.awsErrorDetails().errorMessage().contains("already
exists")) {
                    logger.info("The Security Group already exists. Moving on...");
                } else {
                    logger.error("An unexpected error occurred: {}",
ec2Ex.awsErrorDetails().errorMessage());
                    return;
                }
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);
```

```
        logger.info(DASHES);
        logger.info("4. Display security group information for the new security
group.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
ec2Actions.describeSecurityGroupArnByNameAsync(groupName);
            groupId = future.join();
            logger.info("The security group Id is "+groupId);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                String errorCode = ec2Ex.awsErrorDetails().errorCode();
                if ("InvalidGroup.NotFound".equals(errorCode)) {
                    logger.info("Security group '{}' does not exist. Error Code:
{}", groupName, errorCode);
                } else {
                    logger.info("EC2 error occurred: Message {}, Error Code: {}",
ec2Ex.getMessage(), errorCode);
                }
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Get a list of Amazon Linux 2 AMIs and select one with amzn2
in the name.");
        logger.info("""
            An Amazon EC2 AMI (Amazon Machine Image) is a pre-configured virtual
machine image that
            serves as a template for launching EC2 instances. It contains all the
necessary software and
            configurations required to run an application or operating system on an
EC2 instance.
            """);
        waitForInputToContinue(scanner);
        String instanceAMI="";
        try {
            CompletableFuture<GetParametersByPathResponse> future =
ec2Actions.getParaValuesAsync();
```

```
            GetParametersByPathResponse pathResponse = future.join();
            List<Parameter> parameterList = pathResponse.parameters();
            for (Parameter para : parameterList) {
                if (filterName(para.name())) {
                    instanceAMI = para.value();
                    break;
                }
            }
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        logger.info("The AMI value with amzn2 is: {}", instanceAMI);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. Get the (Amazon Machine Image) AMI value from the amzn2
image.");
        logger.info("""
           An AMI value represents a specific version of a virtual machine (VM) or
server image.
           It uniquely identifies a particular version of an EC2 instance, including
its operating system,
           pre-installed software, and any custom configurations. This allows you to
consistently deploy the same
           VM image across your infrastructure.

            """);
        waitForInputToContinue(scanner);
        String amiValue;
        try {
            CompletableFuture<String> future =
ec2Actions.describeImageAsync(instanceAMI);
            amiValue = future.join();

        } catch (CompletionException ce) {
```

```
            Throwable cause = ce.getCause();
            if (cause instanceof Ec2Exception) {
                Ec2Exception ec2Ex = (Ec2Exception) cause;
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Retrieves an instance type available in the current AWS
region.");
        waitForInputToContinue(scanner);
        String instanceType;
        try {
            CompletableFuture<String> future = ec2Actions.getInstanceTypesAsync();
            instanceType = future.join();
            if (!instanceType.isEmpty()) {
                logger.info("Found instance type: " + instanceType);
            } else {
                logger.info("Desired instance type not found.");
            }
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Create an Amazon EC2 instance using the key pair, the
instance type, the security group, and the EC2 AMI value.");
```

```
        logger.info("Once the EC2 instance is created, it is placed into a running
state.");
        waitForInputToContinue(scanner);
        String newInstanceId;
        try {
            CompletableFuture<String> future =
ec2Actions.runInstanceAsync(instanceType, keyName, groupName, amiValue);
            newInstanceId = future.join();
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception) {
                Ec2Exception ec2Ex = (Ec2Exception) cause;
                switch (ec2Ex.awsErrorDetails().errorCode()) {
                    case "InvalidParameterValue":
                        logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                        break;
                    case "InsufficientInstanceCapacity":
                        // Handle insufficient instance capacity.
                        logger.info("Insufficient instance capacity: {}, {}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                        break;
                    case "InvalidGroup.NotFound":
                        // Handle security group not found.
                        logger.info("Security group not found: {},{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                        break;
                    default:
                        logger.info("EC2 error occurred: {} (Code: {}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                        break;
                }
                return;
            } else {
                logger.info("An unexpected error occurred: {}", (cause != null ?
cause.getMessage() : rt.getMessage()));
                return;
            }
        }
        logger.info("The instance Id is " + newInstanceId);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
```

```
        logger.info("9. Display information about the running instance. ");

        waitForInputToContinue(scanner);
        String publicIp;
        try {
            CompletableFuture<String> future =
ec2Actions.describeEC2InstancesAsync(newInstanceId);
            publicIp = future.join();
            logger.info("EC2 instance public IP {}", publicIp);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }

        }
        logger.info("You can SSH to the instance using this command:");
        logger.info("ssh -i " + fileName + " ec2-user@" + publicIp);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("10. Stop the instance using a waiter (this may take a few
mins).");
        // Remove the 2nd one
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
ec2Actions.stopInstanceAsync(newInstanceId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
```

```
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("11. Start the instance using a waiter (this may take a few
mins).");
        try {
            CompletableFuture<Void> future =
ec2Actions.startInstanceAsync(newInstanceId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                // Handle EC2 exceptions.
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("12. Allocate an Elastic IP address and associate it with the
instance.");
        logger.info("""
            An Elastic IP address is a static public IP address that you can
associate with your EC2 instance.
            This allows you to have a fixed, predictable IP address that remains the
same even if your instance
            is stopped, terminated, or replaced.
            This is particularly useful for applications or services that need to be
accessed consistently from a
            known IP address.
```

```
            An EC2 Allocation ID (also known as a Reserved Instance Allocation
ID) is a unique identifier associated with a Reserved Instance (RI) that you have
purchased in AWS.

            When you purchase a Reserved Instance, AWS assigns a unique Allocation
ID to it.
            This Allocation ID is used to track and identify the specific RI you
have purchased,
            and it is important for managing and monitoring your Reserved Instances.

            """);

        waitForInputToContinue(scanner);
        String allocationId;
        try {
            CompletableFuture<String> future = ec2Actions.allocateAddressAsync();
            allocationId = future.join();
            logger.info("Successfully allocated address with ID: " +allocationId);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        logger.info("The allocation Id value is " + allocationId);
        waitForInputToContinue(scanner);
        String associationId;
        try {
            CompletableFuture<String> future =
ec2Actions.associateAddressAsync(newInstanceId, allocationId);
            associationId = future.join();
            logger.info("Successfully associated address with ID: " +associationId);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
```

```
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("13. Describe the instance again. Note that the public IP
address has changed");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
ec2Actions.describeEC2InstancesAsync(newInstanceId);
            publicIp = future.join();
            logger.info("EC2 instance public IP: " + publicIp);
            logger.info("You can SSH to the instance using this command:");
            logger.info("ssh -i " + fileName + " ec2-user@" + publicIp);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("14. Disassociate and release the Elastic IP address.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<DisassociateAddressResponse> future =
ec2Actions.disassociateAddressAsync(associationId);
            future.join();
            logger.info("Address successfully disassociated.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                // Handle EC2 exceptions.
```

```
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<ReleaseAddressResponse> future =
ec2Actions.releaseEC2AddressAsync(allocationId);
            future.join(); // Wait for the operation to complete
            logger.info("Elastic IP address successfully released.");
        } catch (RuntimeException rte) {
            logger.info("An unexpected error occurred: {}", rte.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("15. Terminate the instance and use a waiter (this may take a
few mins).");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Object> future =
ec2Actions.terminateEC2Async(newInstanceId);
            future.join();
            logger.info("EC2 instance successfully terminated.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                // Handle EC2 exceptions.
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        logger.info(DASHES);
```

```
        logger.info(DASHES);
        logger.info("16. Delete the security group.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
ec2Actions.deleteEC2SecGroupAsync(groupId);
            future.join();
            logger.info("Security group successfully deleted.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("17. Delete the key.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<DeleteKeyPairResponse> future =
ec2Actions.deleteKeysAsync(keyName);
            future.join();
            logger.info("Successfully deleted key pair named " + keyName);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);
```

```
        logger.info(DASHES);
        logger.info("You successfully completed the Amazon EC2 scenario.");
        logger.info(DASHES);
    }
    public static boolean filterName(String name) {
        String[] parts = name.split("/");
        String myValue = parts[4];
        return myValue.contains("amzn2");
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

EC2 작업을 래핑하는 클래스를 정의합니다.

```
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.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.AllocateAddressRequest;
import software.amazon.awssdk.services.ec2.model.AllocateAddressResponse;
import software.amazon.awssdk.services.ec2.model.AssociateAddressRequest;
import software.amazon.awssdk.services.ec2.model.AssociateAddressResponse;
```

```
import
 software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressRequest;
import software.amazon.awssdk.services.ec2.model.CreateKeyPairRequest;
import software.amazon.awssdk.services.ec2.model.CreateKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.DeleteKeyPairRequest;
import software.amazon.awssdk.services.ec2.model.DeleteKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.DeleteSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.DeleteSecurityGroupResponse;
import software.amazon.awssdk.services.ec2.model.DescribeImagesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstanceTypesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstanceTypesResponse;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeKeyPairsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest;
import software.amazon.awssdk.services.ec2.model.DisassociateAddressRequest;
import software.amazon.awssdk.services.ec2.model.DisassociateAddressResponse;
import software.amazon.awssdk.services.ec2.model.DomainType;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.Filter;
import software.amazon.awssdk.services.ec2.model.InstanceTypeInfo;
import software.amazon.awssdk.services.ec2.model.IpPermission;
import software.amazon.awssdk.services.ec2.model.IpRange;
import software.amazon.awssdk.services.ec2.model.ReleaseAddressRequest;
import software.amazon.awssdk.services.ec2.model.ReleaseAddressResponse;
import software.amazon.awssdk.services.ec2.model.RunInstancesRequest;
import software.amazon.awssdk.services.ec2.model.RunInstancesResponse;
import software.amazon.awssdk.services.ec2.model.StopInstancesRequest;
import software.amazon.awssdk.services.ec2.model.StartInstancesRequest;
import software.amazon.awssdk.services.ec2.model.TerminateInstancesRequest;
import software.amazon.awssdk.services.ec2.model.Vpc;
import software.amazon.awssdk.services.ec2.paginators.DescribeImagesPublisher;
import software.amazon.awssdk.services.ec2.paginators.DescribeInstancesPublisher;
import
 software.amazon.awssdk.services.ec2.paginators.DescribeSecurityGroupsPublisher;
import software.amazon.awssdk.services.ec2.paginators.DescribeVpcsPublisher;
import software.amazon.awssdk.services.ec2.waiters.Ec2AsyncWaiter;
import software.amazon.awssdk.services.ssm.SsmAsyncClient;
import software.amazon.awssdk.services.ssm.model.GetParametersByPathRequest;
import software.amazon.awssdk.services.ssm.model.GetParametersByPathResponse;
import software.amazon.awssdk.services.ec2.model.TerminateInstancesResponse;
import java.io.BufferedWriter;
```

```
import java.io.FileWriter;
import java.io.IOException;
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;
import java.util.concurrent.atomic.AtomicReference;

public class EC2Actions {
    private static final Logger logger = LoggerFactory.getLogger(EC2Actions.class);
    private static Ec2AsyncClient ec2AsyncClient;

    /**
     * Retrieves an asynchronous Amazon Elastic Container Registry (ECR) client.
     *
     * @return the configured ECR asynchronous client.
     */
    private static Ec2AsyncClient getAsyncClient() {
        if (ec2AsyncClient == null) {
            /*
            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();
```

```
            ec2AsyncClient = Ec2AsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return ec2AsyncClient;
    }

    /**
     * Deletes a key pair asynchronously.
     *
     * @param keyPair the name of the key pair to delete
     * @return a {@link CompletableFuture} that represents the result of the
asynchronous operation.
     *         The {@link CompletableFuture} will complete with a {@link
DeleteKeyPairResponse} object
     *         that provides the result of the key pair deletion operation.
     */
    public CompletableFuture<DeleteKeyPairResponse> deleteKeysAsync(String keyPair)
{
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
            .keyName(keyPair)
            .build();

        // Initiate the asynchronous request to delete the key pair.
        CompletableFuture<DeleteKeyPairResponse> response =
getAsyncClient().deleteKeyPair(request);
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete key pair: " + keyPair,
ex);
            } else if (resp == null) {
                throw new RuntimeException("No response received for deleting key
pair: " + keyPair);
            }
        });
    }

    /**
     * Deletes an EC2 security group asynchronously.
     *
     * @param groupId the ID of the security group to delete
```

```
     * @return a CompletableFuture that completes when the security group is deleted
     */
    public CompletableFuture<Void> deleteEC2SecGroupAsync(String groupId) {
        DeleteSecurityGroupRequest request = DeleteSecurityGroupRequest.builder()
            .groupId(groupId)
            .build();

        CompletableFuture<DeleteSecurityGroupResponse> response =
getAsyncClient().deleteSecurityGroup(request);
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete security group with Id
" + groupId, ex);
            } else if (resp == null) {
                throw new RuntimeException("No response received for deleting
security group with Id " + groupId);
            }
        }).thenApply(resp -> null);
    }

    /**
     * Terminates an EC2 instance asynchronously and waits for it to reach the
terminated state.
     *
     * @param instanceId the ID of the EC2 instance to terminate
     * @return a {@link CompletableFuture} that completes when the instance has been
terminated
     * @throws RuntimeException if there is no response from the AWS SDK or if there
is a failure during the termination process
     */
    public CompletableFuture<Object> terminateEC2Async(String instanceId) {
        TerminateInstancesRequest terminateRequest =
TerminateInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        CompletableFuture<TerminateInstancesResponse> responseFuture =
getAsyncClient().terminateInstances(terminateRequest);
        return responseFuture.thenCompose(terminateResponse -> {
            if (terminateResponse == null) {
                throw new RuntimeException("No response received for terminating
instance " + instanceId);
            }
```

```
            System.out.println("Going to terminate an EC2 instance and use a waiter
to wait for it to be in terminated state");
            return getAsyncClient().waiter()
                .waitUntilInstanceTerminated(r -> r.instanceIds(instanceId))
                .thenApply(waiterResponse -> null);
        }).exceptionally(throwable -> {
            // Handle any exceptions that occurred during the async call
            throw new RuntimeException("Failed to terminate EC2 instance: " +
throwable.getMessage(), throwable);
        });
    }

    /**
     * Releases an Elastic IP address asynchronously.
     *
     * @param allocId the allocation ID of the Elastic IP address to be released
     * @return a {@link CompletableFuture} representing the asynchronous operation
of releasing the Elastic IP address
     */
    public CompletableFuture<ReleaseAddressResponse> releaseEC2AddressAsync(String
allocId) {
        ReleaseAddressRequest request = ReleaseAddressRequest.builder()
            .allocationId(allocId)
            .build();

        CompletableFuture<ReleaseAddressResponse> response =
getAsyncClient().releaseAddress(request);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to release Elastic IP address",
ex);
            }
        });

        return response;
    }

    /**
     * Disassociates an Elastic IP address from an instance asynchronously.
     *
     * @param associationId The ID of the association you want to disassociate.
     * @return a {@link CompletableFuture} representing the asynchronous operation
of disassociating the address. The
```

```
    *         {@link CompletableFuture} will complete with a {@link
DisassociateAddressResponse} when the operation is
    *         finished.
    * @throws RuntimeException if the disassociation of the address fails.
    */
   public CompletableFuture<DisassociateAddressResponse>
disassociateAddressAsync(String associationId) {
       Ec2AsyncClient ec2 = getAsyncClient();
       DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
           .associationId(associationId)
           .build();

       // Disassociate the address asynchronously.
       CompletableFuture<DisassociateAddressResponse> response =
ec2.disassociateAddress(addressRequest);
       response.whenComplete((resp, ex) -> {
           if (ex != null) {
              throw new RuntimeException("Failed to disassociate address", ex);
           }
       });

       return response;
   }

   /**
    * Associates an Elastic IP address with an EC2 instance asynchronously.
    *
    * @param instanceId    the ID of the EC2 instance to associate the Elastic IP
address with
    * @param allocationId  the allocation ID of the Elastic IP address to associate
    * @return a {@link CompletableFuture} that completes with the association ID
when the operation is successful,
    *         or throws a {@link RuntimeException} if the operation fails
    */
   public CompletableFuture<String> associateAddressAsync(String instanceId, String
allocationId) {
       AssociateAddressRequest associateRequest = AssociateAddressRequest.builder()
           .instanceId(instanceId)
           .allocationId(allocationId)
           .build();

       CompletableFuture<AssociateAddressResponse> responseFuture =
getAsyncClient().associateAddress(associateRequest);
```

```
        return responseFuture.thenApply(response -> {
            if (response.associationId() != null) {
                return response.associationId();
            } else {
                throw new RuntimeException("Association ID is null after associating
address.");
            }
        }).whenComplete((result, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to associate address", ex);
            }
        });
    }

    /**
     * Allocates an Elastic IP address asynchronously in the VPC domain.
     *
     * @return a {@link CompletableFuture} containing the allocation ID of the
allocated Elastic IP address
     */
    public CompletableFuture<String> allocateAddressAsync() {
        AllocateAddressRequest allocateRequest = AllocateAddressRequest.builder()
            .domain(DomainType.VPC)
            .build();

        CompletableFuture<AllocateAddressResponse> responseFuture =
getAsyncClient().allocateAddress(allocateRequest);
        return
responseFuture.thenApply(AllocateAddressResponse::allocationId).whenComplete((result,
ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to allocate address", ex);
            }
        });
    }

    /**
     * Asynchronously describes the state of an EC2 instance.
     * The paginator helps you iterate over multiple pages of results.
     *
     * @param newInstanceId the ID of the EC2 instance to describe
     * @return a {@link CompletableFuture} that, when completed, contains a string
describing the state of the EC2 instance
     */
```

```
   public CompletableFuture<String> describeEC2InstancesAsync(String newInstanceId)
{
       DescribeInstancesRequest request = DescribeInstancesRequest.builder()
           .instanceIds(newInstanceId)
           .build();

       DescribeInstancesPublisher paginator =
getAsyncClient().describeInstancesPaginator(request);
       AtomicReference<String> publicIpAddressRef = new AtomicReference<>();
       return paginator.subscribe(response -> {
           response.reservations().stream()
               .flatMap(reservation -> reservation.instances().stream())
               .filter(instance -> instance.instanceId().equals(newInstanceId))
               .findFirst()
               .ifPresent(instance ->
publicIpAddressRef.set(instance.publicIpAddress()));
       }).thenApply(v -> {
           String publicIpAddress = publicIpAddressRef.get();
           if (publicIpAddress == null) {
               throw new RuntimeException("Instance with ID " + newInstanceId + "
not found.");
           }
           return publicIpAddress;
       }).exceptionally(ex -> {
           logger.info("Failed to describe instances: " + ex.getMessage());
           throw new RuntimeException("Failed to describe instances", ex);
       });
   }

   /**
    * Runs an EC2 instance asynchronously.
    *
    * @param instanceType The instance type to use for the EC2 instance.
    * @param keyName The name of the key pair to associate with the EC2 instance.
    * @param groupName The name of the security group to associate with the EC2
instance.
    * @param amiId The ID of the Amazon Machine Image (AMI) to use for the EC2
instance.
    * @return A {@link CompletableFuture} that completes with the ID of the started
EC2 instance.
    * @throws RuntimeException If there is an error running the EC2 instance.
    */
   public CompletableFuture<String> runInstanceAsync(String instanceType, String
keyName, String groupName, String amiId) {
```

```
        RunInstancesRequest runRequest = RunInstancesRequest.builder()
            .instanceType(instanceType)
            .keyName(keyName)
            .securityGroups(groupName)
            .maxCount(1)
            .minCount(1)
            .imageId(amiId)
            .build();

        CompletableFuture<RunInstancesResponse> responseFuture =
getAsyncClient().runInstances(runRequest);
        return responseFuture.thenCompose(response -> {
            String instanceIdVal = response.instances().get(0).instanceId();
            System.out.println("Going to start an EC2 instance and use a waiter to
wait for it to be in running state");
            return getAsyncClient().waiter()
                .waitUntilInstanceExists(r -> r.instanceIds(instanceIdVal))
                .thenCompose(waitResponse -> getAsyncClient().waiter()
                    .waitUntilInstanceRunning(r -> r.instanceIds(instanceIdVal))
                    .thenApply(runningResponse -> instanceIdVal));
        }).exceptionally(throwable -> {
            // Handle any exceptions that occurred during the async call
            throw new RuntimeException("Failed to run EC2 instance: " +
throwable.getMessage(), throwable);
        });
    }

    /**
     * Asynchronously retrieves the instance types available in the current AWS
region.
     * <p>
     * This method uses the AWS SDK's asynchronous API to fetch the available
instance types
     * and then processes the response. It logs the memory information, network
information,
     * and instance type for each instance type returned. Additionally, it returns a
     * {@link CompletableFuture} that resolves to the instance type string for the
"t2.2xlarge"
     * instance type, if it is found in the response. If the "t2.2xlarge" instance
type is not
     * found, an empty string is returned.
     * </p>
     *
```

```
    * @return a {@link CompletableFuture} that resolves to the instance type string
for the
    * "t2.2xlarge" instance type, or an empty string if the instance type is not
found
    */
   public CompletableFuture<String> getInstanceTypesAsync() {
       DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
           .maxResults(10)
           .build();

       CompletableFuture<DescribeInstanceTypesResponse> response =
getAsyncClient().describeInstanceTypes(typesRequest);
       response.whenComplete((resp, ex) -> {
           if (resp != null) {
               List<InstanceTypeInfo> instanceTypes = resp.instanceTypes();
               for (InstanceTypeInfo type : instanceTypes) {
                   logger.info("The memory information of this type is " +
type.memoryInfo().sizeInMiB());
                   logger.info("Network information is " +
type.networkInfo().toString());
                   logger.info("Instance type is " +
type.instanceType().toString());
               }
           } else {
               throw (RuntimeException) ex;
           }
       });

       return response.thenApply(resp -> {
           for (InstanceTypeInfo type : resp.instanceTypes()) {
               String instanceType = type.instanceType().toString();
               if (instanceType.equals("t2.2xlarge")) {
                   return instanceType;
               }
           }
           return "";
       });
   }

   /**
    * Asynchronously describes an AWS EC2 image with the specified image ID.
    *
    * @param imageId the ID of the image to be described
```

```
    * @return a {@link CompletableFuture} that, when completed, contains the ID of
the described image
    * @throws RuntimeException if no images are found with the provided image ID,
or if an error occurs during the AWS API call
    */
   public CompletableFuture<String> describeImageAsync(String imageId) {
       DescribeImagesRequest imagesRequest = DescribeImagesRequest.builder()
           .imageIds(imageId)
           .build();

       AtomicReference<String> imageIdRef = new AtomicReference<>();
       DescribeImagesPublisher paginator =
getAsyncClient().describeImagesPaginator(imagesRequest);
       return paginator.subscribe(response -> {
           response.images().stream()
               .filter(image -> image.imageId().equals(imageId))
               .findFirst()
               .ifPresent(image -> {
                   logger.info("The description of the image is " +
image.description());
                   logger.info("The name of the image is " + image.name());
                   imageIdRef.set(image.imageId());
               });
       }).thenApply(v -> {
           String id = imageIdRef.get();
           if (id == null) {
               throw new RuntimeException("No images found with the provided image
ID.");
           }
           return id;
       }).exceptionally(ex -> {
           logger.info("Failed to describe image: " + ex.getMessage());
           throw new RuntimeException("Failed to describe image", ex);
       });
   }

   /**
    * Retrieves the parameter values asynchronously using the AWS Systems Manager
(SSM) API.
    *
    * @return a {@link CompletableFuture} that holds the response from the SSM API
call to get parameters by path
    */
   public CompletableFuture<GetParametersByPathResponse> getParaValuesAsync() {
```

```
        SsmAsyncClient ssmClient = SsmAsyncClient.builder()
            .region(Region.US_EAST_1)
            .build();

        GetParametersByPathRequest parameterRequest =
GetParametersByPathRequest.builder()
            .path("/aws/service/ami-amazon-linux-latest")
            .build();

        // Create a CompletableFuture to hold the final result.
        CompletableFuture<GetParametersByPathResponse> responseFuture = new
CompletableFuture<>();
        ssmClient.getParametersByPath(parameterRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    responseFuture.completeExceptionally(new
RuntimeException("Failed to get parameters by path", exception));
                } else {
                    responseFuture.complete(response);
                }
            });

        return responseFuture;
    }


    /**
     * Asynchronously describes the security groups for the specified group ID.
     *
     * @param groupName the name of the security group to describe
     * @return a {@link CompletableFuture} that represents the asynchronous
operation
     *         of describing the security groups. The future will complete with a
     *         {@link DescribeSecurityGroupsResponse} object that contains the
     *         security group information.
     */
    public CompletableFuture<String> describeSecurityGroupArnByNameAsync(String
groupName) {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
            .groupNames(groupName)
            .build();
```

```
        DescribeSecurityGroupsPublisher paginator =
getAsyncClient().describeSecurityGroupsPaginator(request);
        AtomicReference<String> groupIdRef = new AtomicReference<>();
        return paginator.subscribe(response -> {
            response.securityGroups().stream()
                .filter(securityGroup ->
securityGroup.groupName().equals(groupName))
                .findFirst()
                .ifPresent(securityGroup ->
groupIdRef.set(securityGroup.groupId()));
        }).thenApply(v -> {
            String groupId = groupIdRef.get();
            if (groupId == null) {
                throw new RuntimeException("No security group found with the name: "
+ groupName);
            }
            return groupId;
        }).exceptionally(ex -> {
            logger.info("Failed to describe security group: " + ex.getMessage());
            throw new RuntimeException("Failed to describe security group", ex);
        });
    }

    /**
     * Creates a new security group asynchronously with the specified group name,
description, and VPC ID. It also
     * authorizes inbound traffic on ports 80 and 22 from the specified IP address.
     *
     * @param groupName    the name of the security group to create
     * @param groupDesc    the description of the security group
     * @param vpcId        the ID of the VPC in which to create the security group
     * @param myIpAddress  the IP address from which to allow inbound traffic (e.g.,
"192.168.1.1/0" to allow traffic from
     *                     any IP address in the 192.168.1.0/24 subnet)
     * @return a CompletableFuture that, when completed, returns the ID of the
created security group
     * @throws RuntimeException if there was a failure creating the security group
or authorizing the inbound traffic
     */
    public CompletableFuture<String> createSecurityGroupAsync(String groupName,
String groupDesc, String vpcId, String myIpAddress) {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
```

```
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

        return getAsyncClient().createSecurityGroup(createRequest)
            .thenCompose(createResponse -> {
                String groupId = createResponse.groupId();
                IpRange ipRange = IpRange.builder()
                    .cidrIp(myIpAddress + "/32")
                    .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();

                return getAsyncClient().authorizeSecurityGroupIngress(authRequest)
                    .thenApply(authResponse -> groupId);
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    if (exception instanceof CompletionException &&
exception.getCause() instanceof Ec2Exception) {
                        throw (Ec2Exception) exception.getCause();
                    } else {
                        throw new RuntimeException("Failed to create security group:
" + exception.getMessage(), exception);
                    }
                }
```

```
            });
    }

    /**
     * Asynchronously describes the key pairs associated with the current AWS
account.
     *
     * @return a {@link CompletableFuture} containing the {@link
DescribeKeyPairsResponse} object, which provides
     * information about the key pairs.
     */
    public CompletableFuture<DescribeKeyPairsResponse> describeKeysAsync() {
        CompletableFuture<DescribeKeyPairsResponse> responseFuture =
getAsyncClient().describeKeyPairs();
        responseFuture.whenComplete((response, exception) -> {
            if (exception != null) {
              throw new RuntimeException("Failed to describe key pairs: " +
exception.getMessage(), exception);
            }
        });

        return responseFuture;
    }

    /**
     * Creates a new key pair asynchronously.
     *
     * @param keyName the name of the key pair to create
     * @param fileName the name of the file to write the key material to
     * @return a {@link CompletableFuture} that represents the asynchronous
operation
     *         of creating the key pair and writing the key material to a file
     */
    public CompletableFuture<CreateKeyPairResponse> createKeyPairAsync(String
keyName, String fileName) {
        CreateKeyPairRequest request = CreateKeyPairRequest.builder()
            .keyName(keyName)
            .build();

        CompletableFuture<CreateKeyPairResponse> responseFuture =
getAsyncClient().createKeyPair(request);
        responseFuture.whenComplete((response, exception) -> {
            if (response != null) {
                try {
```

```
                    BufferedWriter writer = new BufferedWriter(new
FileWriter(fileName));
                    writer.write(response.keyMaterial());
                    writer.close();
                } catch (IOException e) {
                    throw new RuntimeException("Failed to write key material to
file: " + e.getMessage(), e);
                }
            } else {
                throw new RuntimeException("Failed to create key pair: " +
exception.getMessage(), exception);
            }
        });

        return responseFuture;
    }

    /**
     * Describes the first default VPC asynchronously and using a paginator.
     *
     * @return a {@link CompletableFuture} that, when completed, contains the first
default VPC found.\
     */
    public CompletableFuture<Vpc> describeFirstEC2VpcAsync() {
        Filter myFilter = Filter.builder()
            .name("is-default")
            .values("true")
            .build();

        DescribeVpcsRequest request = DescribeVpcsRequest.builder()
            .filters(myFilter)
            .build();

        DescribeVpcsPublisher paginator =
getAsyncClient().describeVpcsPaginator(request);
        AtomicReference<Vpc> vpcRef = new AtomicReference<>();
        return paginator.subscribe(response -> {
            response.vpcs().stream()
                .findFirst()
                .ifPresent(vpcRef::set);
        }).thenApply(v -> {
            Vpc vpc = vpcRef.get();
            if (vpc == null) {
                throw new RuntimeException("Default VPC not found");
```

```
            }
            return vpc;
        }).exceptionally(ex -> {
            logger.info("Failed to describe VPCs: " + ex.getMessage());
            throw new RuntimeException("Failed to describe VPCs", ex);
        });
    }

    /**
     * Stops the EC2 instance with the specified ID asynchronously and waits for the
instance to stop.
     *
     * @param instanceId the ID of the EC2 instance to stop
     * @return a {@link CompletableFuture} that completes when the instance has been
stopped, or exceptionally if an error occurs
     */
    public CompletableFuture<Void> stopInstanceAsync(String instanceId) {
        StopInstancesRequest stopRequest = StopInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
            .client(getAsyncClient())
            .build();

        CompletableFuture<Void> resultFuture = new CompletableFuture<>();
        logger.info("Stopping instance " + instanceId + " and waiting for it to
stop.");
        getAsyncClient().stopInstances(stopRequest)
            .thenCompose(response -> {
                if (response.stoppingInstances().isEmpty()) {
                    return CompletableFuture.failedFuture(new RuntimeException("No
instances were stopped. Please check the instance ID: " + instanceId));
                }
                return ec2Waiter.waitUntilInstanceStopped(describeRequest);
            })
            .thenAccept(waiterResponse -> {
                logger.info("Successfully stopped instance " + instanceId);
                resultFuture.complete(null);
```

```
            })
            .exceptionally(throwable -> {
                logger.error("Failed to stop instance " + instanceId + ": " +
throwable.getMessage(), throwable);
                resultFuture.completeExceptionally(new RuntimeException("Failed to
stop instance: " + throwable.getMessage(), throwable));
                return null;
            });

        return resultFuture;
    }

    /**
     * Starts an Amazon EC2 instance asynchronously and waits until it is in the
"running" state.
     *
     * @param instanceId the ID of the instance to start
     * @return a {@link CompletableFuture} that completes when the instance has been
started and is in the "running" state, or exceptionally if an error occurs
     */
    public CompletableFuture<Void> startInstanceAsync(String instanceId) {
        StartInstancesRequest startRequest = StartInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
            .client(getAsyncClient())
            .build();

        DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        logger.info("Starting instance " + instanceId + " and waiting for it to
run.");
        CompletableFuture<Void> resultFuture = new CompletableFuture<>();
        return getAsyncClient().startInstances(startRequest)
            .thenCompose(response ->
                ec2Waiter.waitUntilInstanceRunning(describeRequest)
            )
            .thenAccept(waiterResponse -> {
                logger.info("Successfully started instance " + instanceId);
                resultFuture.complete(null);
```

```
            })
            .exceptionally(throwable -> {
                resultFuture.completeExceptionally(new RuntimeException("Failed to
 start instance: " + throwable.getMessage(), throwable));
                return null;
            });
    }

}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 항목을 참조하세요.
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress
  - ReleaseAddress
  - RunInstances
  - StartInstances
  - StopInstances
  - TerminateInstances
  - UnmonitorInstances

작업

`AllocateAddress`

다음 코드 예시는 `AllocateAddress`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Allocates an Elastic IP address asynchronously in the VPC domain.
     *
     * @return a {@link CompletableFuture} containing the allocation ID of the
allocated Elastic IP address
     */
    public CompletableFuture<String> allocateAddressAsync() {
        AllocateAddressRequest allocateRequest = AllocateAddressRequest.builder()
            .domain(DomainType.VPC)
            .build();

        CompletableFuture<AllocateAddressResponse> responseFuture =
getAsyncClient().allocateAddress(allocateRequest);
        return
responseFuture.thenApply(AllocateAddressResponse::allocationId).whenComplete((result,
ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to allocate address", ex);
            }
        });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 AllocateAddress를 참조하세요.

AssociateAddress

다음 코드 예시는 AssociateAddress의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Associates an Elastic IP address with an EC2 instance asynchronously.
    *
    * @param instanceId    the ID of the EC2 instance to associate the Elastic IP
address with
    * @param allocationId  the allocation ID of the Elastic IP address to associate
    * @return a {@link CompletableFuture} that completes with the association ID
when the operation is successful,
    *         or throws a {@link RuntimeException} if the operation fails
    */
   public CompletableFuture<String> associateAddressAsync(String instanceId, String
allocationId) {
       AssociateAddressRequest associateRequest = AssociateAddressRequest.builder()
           .instanceId(instanceId)
           .allocationId(allocationId)
           .build();

       CompletableFuture<AssociateAddressResponse> responseFuture =
getAsyncClient().associateAddress(associateRequest);
       return responseFuture.thenApply(response -> {
           if (response.associationId() != null) {
               return response.associationId();
           } else {
               throw new RuntimeException("Association ID is null after associating
address.");
           }
       }).whenComplete((result, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to associate address", ex);
           }
       });
```

```
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 AssociateAddress를 참조하세요.

`AuthorizeSecurityGroupIngress`

다음 코드 예시는 `AuthorizeSecurityGroupIngress`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Creates a new security group asynchronously with the specified group name,
description, and VPC ID. It also
    * authorizes inbound traffic on ports 80 and 22 from the specified IP address.
    *
    * @param groupName    the name of the security group to create
    * @param groupDesc    the description of the security group
    * @param vpcId        the ID of the VPC in which to create the security group
    * @param myIpAddress  the IP address from which to allow inbound traffic (e.g.,
"192.168.1.1/0" to allow traffic from
    *                     any IP address in the 192.168.1.0/24 subnet)
    * @return a CompletableFuture that, when completed, returns the ID of the
created security group
    * @throws RuntimeException if there was a failure creating the security group
or authorizing the inbound traffic
    */
   public CompletableFuture<String> createSecurityGroupAsync(String groupName,
String groupDesc, String vpcId, String myIpAddress) {
       CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
           .groupName(groupName)
           .description(groupDesc)
           .vpcId(vpcId)
           .build();
```

```
        return getAsyncClient().createSecurityGroup(createRequest)
            .thenCompose(createResponse -> {
                String groupId = createResponse.groupId();
                IpRange ipRange = IpRange.builder()
                    .cidrIp(myIpAddress + "/32")
                    .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();

                return getAsyncClient().authorizeSecurityGroupIngress(authRequest)
                    .thenApply(authResponse -> groupId);
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    if (exception instanceof CompletionException &&
exception.getCause() instanceof Ec2Exception) {
                        throw (Ec2Exception) exception.getCause();
                    } else {
                        throw new RuntimeException("Failed to create security group:
" + exception.getMessage(), exception);
                    }
                }
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 AuthorizeSecurityGroupIngress를 참조하세요.

`CreateKeyPair`

다음 코드 예시는 `CreateKeyPair`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Creates a new key pair asynchronously.
    *
    * @param keyName the name of the key pair to create
    * @param fileName the name of the file to write the key material to
    * @return a {@link CompletableFuture} that represents the asynchronous
operation
    *         of creating the key pair and writing the key material to a file
    */
   public CompletableFuture<CreateKeyPairResponse> createKeyPairAsync(String
keyName, String fileName) {
       CreateKeyPairRequest request = CreateKeyPairRequest.builder()
           .keyName(keyName)
           .build();

       CompletableFuture<CreateKeyPairResponse> responseFuture =
getAsyncClient().createKeyPair(request);
       responseFuture.whenComplete((response, exception) -> {
           if (response != null) {
               try {
                   BufferedWriter writer = new BufferedWriter(new
FileWriter(fileName));
                   writer.write(response.keyMaterial());
                   writer.close();
               } catch (IOException e) {
                   throw new RuntimeException("Failed to write key material to
file: " + e.getMessage(), e);
```

```
                }
            } else {
                throw new RuntimeException("Failed to create key pair: " +
exception.getMessage(), exception);
            }
        });

        return responseFuture;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateKeyPair를 참조하세요.

`CreateSecurityGroup`

다음 코드 예시는 `CreateSecurityGroup`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Creates a new security group asynchronously with the specified group name,
description, and VPC ID. It also
     * authorizes inbound traffic on ports 80 and 22 from the specified IP address.
     *
     * @param groupName    the name of the security group to create
     * @param groupDesc    the description of the security group
     * @param vpcId        the ID of the VPC in which to create the security group
     * @param myIpAddress  the IP address from which to allow inbound traffic (e.g.,
"192.168.1.1/0" to allow traffic from
     *                     any IP address in the 192.168.1.0/24 subnet)
     * @return a CompletableFuture that, when completed, returns the ID of the
created security group
     * @throws RuntimeException if there was a failure creating the security group
or authorizing the inbound traffic
     */
```

```
   public CompletableFuture<String> createSecurityGroupAsync(String groupName,
String groupDesc, String vpcId, String myIpAddress) {
       CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
           .groupName(groupName)
           .description(groupDesc)
           .vpcId(vpcId)
           .build();

       return getAsyncClient().createSecurityGroup(createRequest)
           .thenCompose(createResponse -> {
               String groupId = createResponse.groupId();
               IpRange ipRange = IpRange.builder()
                   .cidrIp(myIpAddress + "/32")
                   .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();

               return getAsyncClient().authorizeSecurityGroupIngress(authRequest)
                   .thenApply(authResponse -> groupId);
           })
           .whenComplete((result, exception) -> {
               if (exception != null) {
                   if (exception instanceof CompletionException &&
exception.getCause() instanceof Ec2Exception) {
                       throw (Ec2Exception) exception.getCause();
```

```
                    } else {
                        throw new RuntimeException("Failed to create security group:
" + exception.getMessage(), exception);
                    }
                }
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateSecurityGroup을 참조하세요.

`DeleteKeyPair`

다음 코드 예시는 `DeleteKeyPair`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Deletes a key pair asynchronously.
     *
     * @param keyPair the name of the key pair to delete
     * @return a {@link CompletableFuture} that represents the result of the
asynchronous operation.
     *         The {@link CompletableFuture} will complete with a {@link
DeleteKeyPairResponse} object
     *         that provides the result of the key pair deletion operation.
     */
    public CompletableFuture<DeleteKeyPairResponse> deleteKeysAsync(String keyPair)
{
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
            .keyName(keyPair)
            .build();

        // Initiate the asynchronous request to delete the key pair.
        CompletableFuture<DeleteKeyPairResponse> response =
getAsyncClient().deleteKeyPair(request);
```

```
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete key pair: " + keyPair,
ex);
            } else if (resp == null) {
                throw new RuntimeException("No response received for deleting key
pair: " + keyPair);
            }
        });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteKeyPair를 참조하세요.

`DeleteSecurityGroup`

다음 코드 예시는 `DeleteSecurityGroup`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Deletes an EC2 security group asynchronously.
     *
     * @param groupId the ID of the security group to delete
     * @return a CompletableFuture that completes when the security group is deleted
     */
    public CompletableFuture<Void> deleteEC2SecGroupAsync(String groupId) {
        DeleteSecurityGroupRequest request = DeleteSecurityGroupRequest.builder()
            .groupId(groupId)
            .build();

        CompletableFuture<DeleteSecurityGroupResponse> response =
getAsyncClient().deleteSecurityGroup(request);
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
```

```
                throw new RuntimeException("Failed to delete security group with Id
" + groupId, ex);
            } else if (resp == null) {
                throw new RuntimeException("No response received for deleting
security group with Id " + groupId);
            }
        }).thenApply(resp -> null);
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteSecurityGroup을 참조하세요.

`DescribeInstanceTypes`

다음 코드 예시는 `DescribeInstanceTypes`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Asynchronously retrieves the instance types available in the current AWS
region.
     * <p>
     * This method uses the AWS SDK's asynchronous API to fetch the available
instance types
     * and then processes the response. It logs the memory information, network
information,
     * and instance type for each instance type returned. Additionally, it returns a
     * {@link CompletableFuture} that resolves to the instance type string for the
"t2.2xlarge"
     * instance type, if it is found in the response. If the "t2.2xlarge" instance
type is not
     * found, an empty string is returned.
     * </p>
     *
     * @return a {@link CompletableFuture} that resolves to the instance type string
for the
```

```
    * "t2.2xlarge" instance type, or an empty string if the instance type is not
found
    */
   public CompletableFuture<String> getInstanceTypesAsync() {
       DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
           .maxResults(10)
           .build();

       CompletableFuture<DescribeInstanceTypesResponse> response =
getAsyncClient().describeInstanceTypes(typesRequest);
       response.whenComplete((resp, ex) -> {
           if (resp != null) {
               List<InstanceTypeInfo> instanceTypes = resp.instanceTypes();
               for (InstanceTypeInfo type : instanceTypes) {
                   logger.info("The memory information of this type is " +
type.memoryInfo().sizeInMiB());
                   logger.info("Network information is " +
type.networkInfo().toString());
                   logger.info("Instance type is " +
type.instanceType().toString());
               }
           } else {
               throw (RuntimeException) ex;
           }
       });

       return response.thenApply(resp -> {
           for (InstanceTypeInfo type : resp.instanceTypes()) {
               String instanceType = type.instanceType().toString();
               if (instanceType.equals("t2.2xlarge")) {
                   return instanceType;
               }
           }
           return "";
       });
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeInstanceTypes를 참조하세요.

DescribeInstances

다음 코드 예시는 DescribeInstances의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Asynchronously describes the state of an EC2 instance.
     * The paginator helps you iterate over multiple pages of results.
     *
     * @param newInstanceId the ID of the EC2 instance to describe
     * @return a {@link CompletableFuture} that, when completed, contains a string
describing the state of the EC2 instance
     */
    public CompletableFuture<String> describeEC2InstancesAsync(String newInstanceId)
{
        DescribeInstancesRequest request = DescribeInstancesRequest.builder()
            .instanceIds(newInstanceId)
            .build();

        DescribeInstancesPublisher paginator =
getAsyncClient().describeInstancesPaginator(request);
        AtomicReference<String> publicIpAddressRef = new AtomicReference<>();
        return paginator.subscribe(response -> {
            response.reservations().stream()
                .flatMap(reservation -> reservation.instances().stream())
                .filter(instance -> instance.instanceId().equals(newInstanceId))
                .findFirst()
                .ifPresent(instance ->
publicIpAddressRef.set(instance.publicIpAddress()));
        }).thenApply(v -> {
            String publicIpAddress = publicIpAddressRef.get();
            if (publicIpAddress == null) {
                throw new RuntimeException("Instance with ID " + newInstanceId + "
not found.");
            }
            return publicIpAddress;
```

```
        }).exceptionally(ex -> {
            logger.info("Failed to describe instances: " + ex.getMessage());
            throw new RuntimeException("Failed to describe instances", ex);
        });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeInstances 참조하세요.

`DescribeKeyPairs`

다음 코드 예시는 `DescribeKeyPairs`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Asynchronously describes the key pairs associated with the current AWS
account.
     *
     * @return a {@link CompletableFuture} containing the {@link
DescribeKeyPairsResponse} object, which provides
     * information about the key pairs.
     */
    public CompletableFuture<DescribeKeyPairsResponse> describeKeysAsync() {
        CompletableFuture<DescribeKeyPairsResponse> responseFuture =
getAsyncClient().describeKeyPairs();
        responseFuture.whenComplete((response, exception) -> {
            if (exception != null) {
              throw new RuntimeException("Failed to describe key pairs: " +
exception.getMessage(), exception);
            }
        });

        return responseFuture;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeKeyPairs를 참조하세요.

`DescribeSecurityGroups`

다음 코드 예시는 `DescribeSecurityGroups`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Asynchronously describes the security groups for the specified group ID.
    *
    * @param groupName the name of the security group to describe
    * @return a {@link CompletableFuture} that represents the asynchronous
operation
    *         of describing the security groups. The future will complete with a
    *         {@link DescribeSecurityGroupsResponse} object that contains the
    *         security group information.
    */
   public CompletableFuture<String> describeSecurityGroupArnByNameAsync(String
groupName) {
       DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
           .groupNames(groupName)
           .build();

       DescribeSecurityGroupsPublisher paginator =
getAsyncClient().describeSecurityGroupsPaginator(request);
       AtomicReference<String> groupIdRef = new AtomicReference<>();
       return paginator.subscribe(response -> {
           response.securityGroups().stream()
               .filter(securityGroup ->
securityGroup.groupName().equals(groupName))
```

```
                .findFirst()
                .ifPresent(securityGroup ->
groupIdRef.set(securityGroup.groupId()));
        }).thenApply(v -> {
            String groupId = groupIdRef.get();
            if (groupId == null) {
                throw new RuntimeException("No security group found with the name: "
+ groupName);
            }
            return groupId;
        }).exceptionally(ex -> {
            logger.info("Failed to describe security group: " + ex.getMessage());
            throw new RuntimeException("Failed to describe security group", ex);
        });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeSecurityGroups를 참조하세요.

`DisassociateAddress`

다음 코드 예시는 `DisassociateAddress`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Disassociates an Elastic IP address from an instance asynchronously.
     *
     * @param associationId The ID of the association you want to disassociate.
     * @return a {@link CompletableFuture} representing the asynchronous operation
of disassociating the address. The
     *         {@link CompletableFuture} will complete with a {@link
DisassociateAddressResponse} when the operation is
     *         finished.
     * @throws RuntimeException if the disassociation of the address fails.
     */
```

```
    public CompletableFuture<DisassociateAddressResponse>
disassociateAddressAsync(String associationId) {
        Ec2AsyncClient ec2 = getAsyncClient();
        DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
            .associationId(associationId)
            .build();

        // Disassociate the address asynchronously.
        CompletableFuture<DisassociateAddressResponse> response =
ec2.disassociateAddress(addressRequest);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to disassociate address", ex);
            }
        });

        return response;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DisassociateAddress를 참조하세요.

`GetPasswordData`

다음 코드 예시는 `GetPasswordData`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.*;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetPasswordData {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                   <instanceId>

                Where:
                   instanceId - An instance id value that you can obtain from the
 AWS Management Console.\s
             """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
        String instanceId = args[0];
        Ec2AsyncClient ec2AsyncClient = Ec2AsyncClient.builder()
            .region(Region.US_EAST_1)
            .build();

        try {
            CompletableFuture<Void> future = getPasswordDataAsync(ec2AsyncClient,
 instanceId);
            future.join();
        } catch (RuntimeException rte) {
            System.err.println("An exception occurred: " + (rte.getCause() != null ?
 rte.getCause().getMessage() : rte.getMessage()));
        }
    }

    /**
     * Fetches the password data for the specified EC2 instance asynchronously.
     *
     * @param ec2AsyncClient the EC2 asynchronous client to use for the request
     * @param instanceId instanceId the ID of the EC2 instance for which you want to
 fetch the password data
```

```
     * @return a {@link CompletableFuture} that completes when the password data has
 been fetched
     * @throws RuntimeException if there was a failure in fetching the password data
     */
    public static CompletableFuture<Void> getPasswordDataAsync(Ec2AsyncClient
 ec2AsyncClient, String instanceId) {
        GetPasswordDataRequest getPasswordDataRequest =
 GetPasswordDataRequest.builder()
            .instanceId(instanceId)
            .build();


        CompletableFuture<GetPasswordDataResponse> response =
 ec2AsyncClient.getPasswordData(getPasswordDataRequest);
        response.whenComplete((getPasswordDataResponse, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to get password data for
 instance: " + instanceId, ex);
            } else if (getPasswordDataResponse == null ||
 getPasswordDataResponse.passwordData().isEmpty()) {
                throw new RuntimeException("No password data found for instance: " +
 instanceId);
            } else {
                String encryptedPasswordData =
 getPasswordDataResponse.passwordData();
                System.out.println("Encrypted Password Data: " +
 encryptedPasswordData);
            }
        });

        return response.thenApply(resp -> null);
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetPasswordData를 참조하세요.

`ReleaseAddress`

다음 코드 예시는 `ReleaseAddress`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Releases an Elastic IP address asynchronously.
    *
    * @param allocId the allocation ID of the Elastic IP address to be released
    * @return a {@link CompletableFuture} representing the asynchronous operation
of releasing the Elastic IP address
    */
   public CompletableFuture<ReleaseAddressResponse> releaseEC2AddressAsync(String
allocId) {
       ReleaseAddressRequest request = ReleaseAddressRequest.builder()
           .allocationId(allocId)
           .build();

       CompletableFuture<ReleaseAddressResponse> response =
getAsyncClient().releaseAddress(request);
       response.whenComplete((resp, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to release Elastic IP address",
ex);
           }
       });

       return response;
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ReleaseAddress를 참조하세요.

`RunInstances`

다음 코드 예시는 `RunInstances`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Runs an EC2 instance asynchronously.
    *
    * @param instanceType The instance type to use for the EC2 instance.
    * @param keyName The name of the key pair to associate with the EC2 instance.
    * @param groupName The name of the security group to associate with the EC2
instance.
    * @param amiId The ID of the Amazon Machine Image (AMI) to use for the EC2
instance.
    * @return A {@link CompletableFuture} that completes with the ID of the started
EC2 instance.
    * @throws RuntimeException If there is an error running the EC2 instance.
    */
   public CompletableFuture<String> runInstanceAsync(String instanceType, String
keyName, String groupName, String amiId) {
       RunInstancesRequest runRequest = RunInstancesRequest.builder()
           .instanceType(instanceType)
           .keyName(keyName)
           .securityGroups(groupName)
           .maxCount(1)
           .minCount(1)
           .imageId(amiId)
           .build();

       CompletableFuture<RunInstancesResponse> responseFuture =
getAsyncClient().runInstances(runRequest);
       return responseFuture.thenCompose(response -> {
           String instanceIdVal = response.instances().get(0).instanceId();
           System.out.println("Going to start an EC2 instance and use a waiter to
wait for it to be in running state");
           return getAsyncClient().waiter()
               .waitUntilInstanceExists(r -> r.instanceIds(instanceIdVal))
               .thenCompose(waitResponse -> getAsyncClient().waiter()
                   .waitUntilInstanceRunning(r -> r.instanceIds(instanceIdVal))
```

```
                    .thenApply(runningResponse -> instanceIdVal));
        }).exceptionally(throwable -> {
            // Handle any exceptions that occurred during the async call
            throw new RuntimeException("Failed to run EC2 instance: " +
throwable.getMessage(), throwable);
        });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 RunInstances를 참조하세요.

`StartInstances`

다음 코드 예시는 `StartInstances`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Starts an Amazon EC2 instance asynchronously and waits until it is in the
"running" state.
     *
     * @param instanceId the ID of the instance to start
     * @return a {@link CompletableFuture} that completes when the instance has been
started and is in the "running" state, or exceptionally if an error occurs
     */
    public CompletableFuture<Void> startInstanceAsync(String instanceId) {
        StartInstancesRequest startRequest = StartInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
            .client(getAsyncClient())
            .build();

        DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
```

```
            .instanceIds(instanceId)
            .build();

        logger.info("Starting instance " + instanceId + " and waiting for it to
run.");
        CompletableFuture<Void> resultFuture = new CompletableFuture<>();
        return getAsyncClient().startInstances(startRequest)
            .thenCompose(response ->
                ec2Waiter.waitUntilInstanceRunning(describeRequest)
            )
            .thenAccept(waiterResponse -> {
                logger.info("Successfully started instance " + instanceId);
                resultFuture.complete(null);
            })
            .exceptionally(throwable -> {
                resultFuture.completeExceptionally(new RuntimeException("Failed to
start instance: " + throwable.getMessage(), throwable));
                return null;
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartInstances를 참조하세요.

`StopInstances`

다음 코드 예시는 `StopInstances`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Stops the EC2 instance with the specified ID asynchronously and waits for the
instance to stop.
     *
     * @param instanceId the ID of the EC2 instance to stop
```

```
     * @return a {@link CompletableFuture} that completes when the instance has been
stopped, or exceptionally if an error occurs
     */
    public CompletableFuture<Void> stopInstanceAsync(String instanceId) {
        StopInstancesRequest stopRequest = StopInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
            .client(getAsyncClient())
            .build();

        CompletableFuture<Void> resultFuture = new CompletableFuture<>();
        logger.info("Stopping instance " + instanceId + " and waiting for it to
stop.");
        getAsyncClient().stopInstances(stopRequest)
            .thenCompose(response -> {
                if (response.stoppingInstances().isEmpty()) {
                    return CompletableFuture.failedFuture(new RuntimeException("No
instances were stopped. Please check the instance ID: " + instanceId));
                }
                return ec2Waiter.waitUntilInstanceStopped(describeRequest);
            })
            .thenAccept(waiterResponse -> {
                logger.info("Successfully stopped instance " + instanceId);
                resultFuture.complete(null);
            })
            .exceptionally(throwable -> {
                logger.error("Failed to stop instance " + instanceId + ": " +
throwable.getMessage(), throwable);
                resultFuture.completeExceptionally(new RuntimeException("Failed to
stop instance: " + throwable.getMessage(), throwable));
                return null;
            });

        return resultFuture;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StopInstances를 참조하세요.

`TerminateInstances`

다음 코드 예시는 `TerminateInstances`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Terminates an EC2 instance asynchronously and waits for it to reach the
terminated state.
    *
    * @param instanceId the ID of the EC2 instance to terminate
    * @return a {@link CompletableFuture} that completes when the instance has been
terminated
    * @throws RuntimeException if there is no response from the AWS SDK or if there
is a failure during the termination process
    */
   public CompletableFuture<Object> terminateEC2Async(String instanceId) {
       TerminateInstancesRequest terminateRequest =
TerminateInstancesRequest.builder()
           .instanceIds(instanceId)
           .build();

       CompletableFuture<TerminateInstancesResponse> responseFuture =
getAsyncClient().terminateInstances(terminateRequest);
       return responseFuture.thenCompose(terminateResponse -> {
           if (terminateResponse == null) {
               throw new RuntimeException("No response received for terminating
instance " + instanceId);
           }
           System.out.println("Going to terminate an EC2 instance and use a waiter
to wait for it to be in terminated state");
           return getAsyncClient().waiter()
               .waitUntilInstanceTerminated(r -> r.instanceIds(instanceId))
               .thenApply(waiterResponse -> null);
```

```
        }).exceptionally(throwable -> {
            // Handle any exceptions that occurred during the async call
            throw new RuntimeException("Failed to terminate EC2 instance: " +
throwable.getMessage(), throwable);
        });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 TerminateInstances를 참조하세요.

시나리오

복원력이 뛰어난 서비스 구축 및 관리

다음 코드 예제에서는 책, 영화, 노래 추천을 반환하는 로드 밸런싱 웹 서비스를 만드는 방법을 보여줍니다. 이 예제에서는 서비스가 장애에 대응하는 방법과 장애 발생 시 복원력을 높이기 위해 서비스를 재구성하는 방법을 보여줍니다.

- Amazon EC2 Auto Scaling 그룹을 사용하여 시작 템플릿을 기반으로 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스를 생성하고 인스턴스 수를 지정된 범위 내로 유지합니다.
- Elastic Load Balancing으로 HTTP 요청을 처리하고 배포합니다.
- Auto Scaling 그룹의 인스턴스 상태를 모니터링하고 요청을 정상 인스턴스로만 전달합니다.
- 각 EC2 인스턴스에서 Python 웹 서버를 실행하여 HTTP 요청을 처리합니다. 웹 서버는 추천 및 상태 확인으로 응답합니다.
- Amazon DynamoDB 테이블을 사용하여 추천 서비스를 시뮬레이션합니다.
- AWS Systems Manager 파라미터를 업데이트하여 요청 및 상태 확인에 대한 웹 서버 응답을 제어합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
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);
    }
}
```

Auto Scaling과 Amazon EC2 작업을 래핑하는 클래스를 생성합니다.

```
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.");
    }
}
```

ELB 작업을 래핑하는 클래스를 생성합니다.

```
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)
                    .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 "";
    }
}
```

DynamoDB를 사용하여 추천 서비스를 시뮬레이션하는 클래스를 생성합니다.

```
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);
    }
}
```

Systems Manager 작업을 래핑하는 클래스를 생성합니다.

```
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);
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - 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

SDK for Java 2.x를 사용한 Amazon ECR 예제

다음 코드 예제에서는 AWS SDK for Java 2.x Amazon ECR에서를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시작하기

Hello Amazon ECR

다음 코드 예제에서는 Amazon ECR 사용을 시작하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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()));
    }
}
```

- API에 대한 세부 정보는AWS SDK for Java 2.x API 참조의 listImages를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- Amazon ECR 리포지토리를 생성합니다.
- 리포지토리 정책을 설정합니다.
- 리포지토리 URI를 검색합니다.
- Amazon ECR 인증 토큰을 가져옵니다.
- Amazon ECR 리포지토리의 수명 주기 정책을 설정합니다.
- Amazon ECR 리포지토리에 Docker 이미지를 푸시합니다.
- Amazon ECR 리포지토리에 이미지가 있는지 확인합니다.

- 계정의 Amazon ECR 리포지토리를 나열하고 관련 세부 정보를 가져옵니다.
- Amazon ECR 리포지토리를 삭제합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon ECR 기능을 시연하는 대화형 시나리오를 실행합니다.

```
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:
 *
 * /scenarios/basics/ecr/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.");
           }
       }
   }
}
```

Amazon ECR SDK 메서드의 래퍼 클래스입니다.

```
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;
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

작업

`CreateRepository`

다음 코드 예시는 `CreateRepository`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * 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);
```

```
            }
        }
    }
```

- API 세부 정보는AWS SDK for Java 2.x API 참조의 CreateRepository를 참조하세요.

`DeleteRepository`

다음 코드 예시는 `DeleteRepository`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * 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();
    }
```

- API 세부 정보는AWS SDK for Java 2.x API 참조의 DeleteRepository를 참조하세요.

`DescribeImages`

다음 코드 예시는 `DescribeImages`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * 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();
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeImages를 참조하세요.

`DescribeRepositories`

다음 코드 예시는 `DescribeRepositories`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * 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();
    }
```

- API 세부 정보는AWS SDK for Java 2.x API 참조의 DescribeRepositories를 참조하세요.

`GetAuthorizationToken`

다음 코드 예시는 `GetAuthorizationToken`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * 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();
   }
```

- API 세부 정보는AWS SDK for Java 2.x API 참조의 GetAuthorizationToken을 참조하세요.

GetRepositoryPolicy

다음 코드 예시는 GetRepositoryPolicy의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * 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;
    }
```

- API 세부 정보는AWS SDK for Java 2.x API 참조의 GetRepositoryPolicy를 참조하세요.

`PushImageCmd`

다음 코드 예시는 `PushImageCmd`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * 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();
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PushImageCmd를 참조하세요.

SetRepositoryPolicy

다음 코드 예시는 SetRepositoryPolicy의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * 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();
    }
```

- API 세부 정보는AWS SDK for Java 2.x API 참조의 SetRepositoryPolicy를 참조하세요.

`StartLifecyclePolicyPreview`

다음 코드 예시는 `StartLifecyclePolicyPreview`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
```

```
    * 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();
   }
```

- API 세부 정보는AWS SDK for Java 2.x API 참조의 StartLifecyclePolicyPreview를 참조하세요.

Java 2.x용 SDK를 사용하는 Amazon ECS 예제

다음 코드 예제에서는 AWS SDK for Java 2.x Amazon ECS에서를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`CreateCluster`

다음 코드 예시는 `CreateCluster`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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 "";
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateCluster를 참조하세요.

`CreateService`

다음 코드 예시는 `CreateService`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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 "";
        }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateService를 참조하세요.

`DeleteService`

다음 코드 예시는 `DeleteService`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteService를 참조하세요.

`DescribeClusters`

다음 코드 예시는 `DescribeClusters`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeClusters를 참조하세요.

`DescribeTasks`

다음 코드 예시는 `DescribeTasks`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeTasks를 참조하세요.

ListClusters

다음 코드 예시는 ListClusters의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListClusters를 참조하세요.

`UpdateService`

다음 코드 예시는 `UpdateService`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateService를 참조하세요.

ELB - SDK for Java 2.x를 사용한 버전 2 예제

다음 코드 예제에서는 ELB - 버전 2와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시작하기
- 작업
- 시나리오

시작하기

Hello ELB

다음 코드 예제에서는 ELB 사용을 시작하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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());
        }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeLoadBalancers를 참조하세요.

작업

`CreateListener`

다음 코드 예시는 `CreateListener`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /*
     * 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)
                    .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 "";
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateListener를 참조하세요.

`CreateLoadBalancer`

다음 코드 예시는 `CreateLoadBalancer`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /*
     * 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)
                    .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 "";
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateLoadBalancer를 참조하세요.

`CreateTargetGroup`

다음 코드 예시는 `CreateTargetGroup`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /*
     * 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;
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateTargetGroup를 참조하세요.

`DeleteLoadBalancer`

다음 코드 예시는 `DeleteLoadBalancer`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteLoadBalancer를 참조하세요.

`DeleteTargetGroup`

다음 코드 예시는 `DeleteTargetGroup`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteTargetGroup을 참조하세요.

`DescribeTargetHealth`

다음 코드 예시는 `DescribeTargetHealth`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    // 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();
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeTargetHealth를 참조하세요.

시나리오

복원력이 뛰어난 서비스 구축 및 관리

다음 코드 예제에서는 책, 영화, 노래 추천을 반환하는 로드 밸런싱 웹 서비스를 만드는 방법을 보여줍니다. 이 예제에서는 서비스가 장애에 대응하는 방법과 장애 발생 시 복원력을 높이기 위해 서비스를 재구성하는 방법을 보여줍니다.

- Amazon EC2 Auto Scaling 그룹을 사용하여 시작 템플릿을 기반으로 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스를 생성하고 인스턴스 수를 지정된 범위 내로 유지합니다.
- Elastic Load Balancing으로 HTTP 요청을 처리하고 배포합니다.
- Auto Scaling 그룹의 인스턴스 상태를 모니터링하고 요청을 정상 인스턴스로만 전달합니다.
- 각 EC2 인스턴스에서 Python 웹 서버를 실행하여 HTTP 요청을 처리합니다. 웹 서버는 추천 및 상태 확인으로 응답합니다.
- Amazon DynamoDB 테이블을 사용하여 추천 서비스를 시뮬레이션합니다.
- AWS Systems Manager 파라미터를 업데이트하여 요청 및 상태 확인에 대한 웹 서버 응답을 제어합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
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);
    }
}
```

Auto Scaling과 Amazon EC2 작업을 래핑하는 클래스를 생성합니다.

```
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.");
    }
}
```

ELB 작업을 래핑하는 클래스를 생성합니다.

```
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)
                    .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 "";
    }
}
```

DynamoDB를 사용하여 추천 서비스를 시뮬레이션하는 클래스를 생성합니다.

```
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);
    }
}
```

Systems Manager 작업을 래핑하는 클래스를 생성합니다.

```
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);
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - 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

SDK for Java 2.x를 사용한 MediaStore 예시

다음 코드 예제에서는 MediaStore와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`CreateContainer`

다음 코드 예시는 `CreateContainer`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateContainer를 참조하세요.

`DeleteContainer`

다음 코드 예시는 `DeleteContainer`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteContainer를 참조하세요.

`DeleteObject`

다음 코드 예시는 `DeleteObject`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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();
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteObject를 참조하세요.

`DescribeContainer`

다음 코드 예시는 `DescribeContainer`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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 "";
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeContainer를 참조하세요.

`GetObject`

다음 코드 예시는 `GetObject`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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();
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetObject를 참조하세요.

`ListContainers`

다음 코드 예시는 `ListContainers`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListContainers를 참조하세요.

`PutObject`

다음 코드 예시는 `PutObject`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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();
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutObject를 참조하세요.

AWS Entity Resolution SDK for Java 2.x를 사용한 예제

다음 코드 예제에서는와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS Entity Resolution.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시작하기

안녕하세요 AWS Entity Resolution

다음 코드 예제에서는 AWS Entity Resolution를 사용하여 시작하는 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloEntityResoultion {

    private static final Logger logger =
 LoggerFactory.getLogger(HelloEntityResoultion.class);
```

```
   private static EntityResolutionAsyncClient entityResolutionAsyncClient;
   public static void main(String[] args) {
       listMatchingWorkflows();
   }

   public static EntityResolutionAsyncClient getResolutionAsyncClient() {
       if (entityResolutionAsyncClient == null) {
           /*
           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.
               .retryStrategy(RetryMode.STANDARD)
               .build();

           entityResolutionAsyncClient = EntityResolutionAsyncClient.builder()
               .httpClient(httpClient)
               .overrideConfiguration(overrideConfig)
               .build();
       }
       return entityResolutionAsyncClient;
   }
```

```
    /**
     * Lists all matching workflows using an asynchronous paginator.
     * <p>
     * This method requests a paginated list of matching workflows from the
     * AWS Entity Resolution service and logs the names of the retrieved workflows.
     * It uses an asynchronous approach with a paginator and waits for the operation
     * to complete using {@code CompletableFuture#join()}.
     * </p>
     */
    public static void listMatchingWorkflows() {
        ListMatchingWorkflowsRequest request =
 ListMatchingWorkflowsRequest.builder().build();

        ListMatchingWorkflowsPublisher paginator =
            getResolutionAsyncClient().listMatchingWorkflowsPaginator(request);

        // Iterate through the paginated results asynchronously
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            response.workflowSummaries().forEach(workflow ->
                logger.info("Matching Workflow Name: " + workflow.workflowName())
            );
        });

        // Wait for the asynchronous operation to complete
        future.join();
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListMatchingWorkflows를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 스키마 매핑을 생성합니다.
- AWS Entity Resolution 워크플로를 생성합니다.
- 워크플로에 대해 일치하는 작업을 시작합니다.
- 일치하는 작업에 대한 세부 정보를 가져옵니다.

- 스키마 매핑을 가져옵니다.
- 모든 스키마 매핑을 나열합니다.
- 스키마 매핑 리소스에 태그를 지정합니다.
- AWS Entity Resolution 자산을 삭제합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

AWS Entity Resolution 기능을 보여주는 대화형 시나리오를 실행합니다.

```
public class EntityResScenario {
    private static final Logger logger =
 LoggerFactory.getLogger(EntityResScenario.class);
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final String STACK_NAME = "EntityResolutionCdkStack2";
    private static final String ENTITY_RESOLUTION_ROLE_ARN_KEY =
 "EntityResolutionRoleArn";
    private static final String GLUE_DATA_BUCKET_NAME_KEY = "GlueDataBucketName";
    private static final String JSON_GLUE_TABLE_ARN_KEY = "JsonErGlueTableArn";
    private static final String CSV_GLUE_TABLE_ARN_KEY = "CsvErGlueTableArn";
    private static String glueBucketName;
    private static String workflowName = "workflow-" + UUID.randomUUID();

    private static String jsonSchemaMappingName = "jsonschema-" + UUID.randomUUID();
    private static String jsonSchemaMappingArn = null;
    private static String csvSchemaMappingName = "csv-" + UUID.randomUUID();
    private static String roleARN;
    private static String csvGlueTableArn;
    private static String jsonGlueTableArn;
    private static Scanner scanner = new Scanner(System.in);

    private static EntityResActions actions = new EntityResActions();

    public static void main(String[] args) throws InterruptedException {

        logger.info("Welcome to the AWS Entity Resolution Scenario.");
```

```
        logger.info("""
            AWS Entity Resolution is a fully-managed machine learning service
provided by
            Amazon Web Services (AWS) that helps organizations extract, link, and
            organize information from multiple data sources. It leverages natural
            language processing and deep learning models to identify and resolve
            entities, such as people, places, organizations, and products,
            across structured and unstructured data.

            With Entity Resolution, customers can build robust data integration
            pipelines to combine and reconcile data from multiple systems,
databases,
            and documents. The service can handle ambiguous, incomplete, or
conflicting
            information, and provide a unified view of entities and their
relationships.
            This can be particularly valuable in applications such as customer 360,
            fraud detection, supply chain management, and knowledge management,
where
            accurate entity identification is crucial.

            The `EntityResolutionAsyncClient` interface in the AWS SDK for Java 2.x
            provides a set of methods to programmatically interact with the AWS
Entity
            Resolution service. This allows developers to automate the entity
extraction,
            linking, and deduplication process as part of their data processing
workflows.
            With Entity Resolution, organizations can unlock the value of their
data,
            improve decision-making, and enhance customer experiences by having a
reliable,
            comprehensive view of their key entities.
            """);

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("""
            To prepare the AWS resources needed for this scenario application, the
next step uploads
            a CloudFormation template whose resulting stack creates the following
resources:
```

```
            - An AWS Glue Data Catalog table
            - An AWS IAM role
            - An AWS S3 bucket
            - An AWS Entity Resolution Schema

            It can take a couple minutes for the Stack to finish creating the
resources.
            """);
        waitForInputToContinue(scanner);
        logger.info("Generating resources...");
        CloudFormationHelper.deployCloudFormationStack(STACK_NAME);
        Map<String, String> outputsMap =
CloudFormationHelper.getStackOutputsAsync(STACK_NAME).join();
        roleARN = outputsMap.get(ENTITY_RESOLUTION_ROLE_ARN_KEY);
        glueBucketName = outputsMap.get(GLUE_DATA_BUCKET_NAME_KEY);
        csvGlueTableArn = outputsMap.get(CSV_GLUE_TABLE_ARN_KEY);
        jsonGlueTableArn = outputsMap.get(JSON_GLUE_TABLE_ARN_KEY);
        logger.info(DASHES);
        waitForInputToContinue(scanner);

        try {
            runScenario();

        } catch (Exception ce) {
            Throwable cause = ce.getCause();
            logger.error("An exception happened: " + (cause != null ?
cause.getMessage() : ce.getMessage()));
        }
    }

    private static void runScenario() throws InterruptedException {
        /*
         This JSON is a valid input for the AWS Entity Resolution service.
         The JSON represents an array of three objects, each containing an "id",
"name", and "email"
         property. This format aligns with the expected input structure for the
         Entity Resolution service.
         */
        String json = """
            {"id":"1","name":"Jane Doe","email":"jane.doe@example.com"}
            {"id":"2","name":"John Doe","email":"john.doe@example.com"}
            {"id":"3","name":"Jorge Souza","email":"jorge_souza@example.com"}
            """;
```

```
        logger.info("Upload the following JSON objects to the {} S3 bucket.",
 glueBucketName);
        logger.info(json);
        String csv = """
            id,name,email,phone
            1,Jane B.,Doe,jane.doe@example.com,555-876-9846
            2,John Doe Jr.,john.doe@example.com,555-654-3210
            3,María García,maría_garcia@company.com,555-567-1234
            4,Mary Major,mary_major@company.com,555-222-3333
            """;
        logger.info("Upload the following CSV data to the {} S3 bucket.",
 glueBucketName);
        logger.info(csv);
        waitForInputToContinue(scanner);
        try {
            actions.uploadInputData(glueBucketName, json, csv);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();

            if (cause == null) {
                logger.error("Failed to upload input data: {}", ce.getMessage(),
 ce);
            }

            if (cause instanceof ResourceNotFoundException) {
                logger.error("Failed to upload input data as the resource was not
 found: {}", cause.getMessage(), cause);
            }
            return;
        }
        logger.info("The JSON and CSV objects have been uploaded to the S3
 bucket.");
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("1. Create Schema Mapping");
        logger.info("""
            Entity Resolution schema mapping aligns and integrates data from
            multiple sources by identifying and matching corresponding entities
            like customers or products. It unifies schemas, resolves conflicts,
            and uses machine learning to link related entities, enabling a
            consolidated, accurate view for improved data quality and decision-
making.
```

```
            In this example, the schema mapping lines up with the fields in the JSON
and CSV objects. That is,
            it contains these fields: id, name, and email.
            """);
        try {
            CreateSchemaMappingResponse response =
actions.createSchemaMappingAsync(jsonSchemaMappingName).join();
            jsonSchemaMappingName = response.schemaName();
            logger.info("The JSON schema mapping name is " + jsonSchemaMappingName);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();

            if (cause == null) {
                logger.error("Failed to create JSON schema mapping: {}",
ce.getMessage(), ce);
            }

            if (cause instanceof ConflictException) {
                logger.error("Schema mapping conflict detected: {}",
cause.getMessage(), cause);
            } else {
                logger.error("Unexpected error while creating schema mapping: {}",
cause.getMessage(), cause);
            }
            return;
        }

        try {
            CreateSchemaMappingResponse response =
actions.createSchemaMappingAsync(csvSchemaMappingName).join();
            csvSchemaMappingName = response.schemaName();
            logger.info("The CSV schema mapping name is " + csvSchemaMappingName);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause == null) {
                logger.error("Failed to create CSV schema mapping: {}",
ce.getMessage(), ce);
            }

            if (cause instanceof ConflictException) {
                logger.error("Schema mapping conflict detected: {}",
cause.getMessage(), cause);
            } else {
```

```
                logger.error("Unexpected error while creating CSV schema mapping:
{}", cause.getMessage(), cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Create an AWS Entity Resolution Workflow. ");
        logger.info("""
            An Entity Resolution matching workflow identifies and links records
            across datasets that represent the same real-world entity, such as
            customers or products. Using techniques like schema mapping,
            data profiling, and machine learning algorithms,
            it evaluates attributes like names or emails to detect duplicates
            or relationships, even with variations or inconsistencies.
            The workflow outputs consolidated, de-duplicated data.

            We will use the machine learning-based matching technique.
            """);
        waitForInputToContinue(scanner);
        try {
            String workflowArn = actions.createMatchingWorkflowAsync(
                roleARN, workflowName, glueBucketName, jsonGlueTableArn,
                jsonSchemaMappingName, csvGlueTableArn,
csvSchemaMappingName).join();

            logger.info("The workflow ARN is: " + workflowArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();

            if (cause == null) {
                logger.error("An unexpected error occurred: {}", ce.getMessage(),
ce);
            }

            if (cause instanceof ValidationException) {
                logger.error("Validation error: {}", cause.getMessage(), cause);
            } else if (cause instanceof ConflictException) {
                logger.error("Workflow conflict detected: {}", cause.getMessage(),
cause);
            } else {
                logger.error("Unexpected error: {}", cause.getMessage(), cause);
```

```
            }
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);
        logger.info("3. Start the matching job of the " + workflowName + "
workflow.");
        waitForInputToContinue(scanner);
        String jobId = null;
        try {
            jobId = actions.startMatchingJobAsync(workflowName).join();
            logger.info("The matching job was successfully started.");
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ConflictException) {
                logger.error("Job conflict detected: {}", cause.getMessage(),
cause);
            } else {
                logger.error("Unexpected error while starting the job: {}",
ce.getMessage(), ce);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. While the matching job is running, let's look at other API
methods. First, let's get details for job " + jobId);
        waitForInputToContinue(scanner);
        try {
            actions.getMatchingJobAsync(jobId, workflowName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The matching job not found: {}", cause.getMessage(),
cause);
            } else {
                logger.error("Failed to start matching job: " + (cause != null ?
cause.getMessage() : ce.getMessage()));
            }
            return;
        }
```

```
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Get the schema mapping for the JSON data.");
        waitForInputToContinue(scanner);
        try {
            GetSchemaMappingResponse response =
actions.getSchemaMappingAsync(jsonSchemaMappingName).join();
            jsonSchemaMappingArn = response.schemaArn();
            logger.info("Schema mapping ARN is " + jsonSchemaMappingArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("Schema mapping not found: {}", cause.getMessage(),
cause);
            } else {
                logger.error("Error retrieving the specific schema mapping: " +
ce.getCause().getMessage());
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. List Schema Mappings.");
        try {
            actions.ListSchemaMappings();
        } catch (CompletionException ce) {
            logger.error("Error retrieving schema mappings: " +
ce.getCause().getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Tag the {} resource.", jsonSchemaMappingName);
        logger.info("""
            Tags can help you organize and categorize your Entity Resolution
resources.
            You can also use them to scope user permissions by granting a user
permission
            to access or change only resources with certain tag values.
```

```
            In Entity Resolution, SchemaMapping and MatchingWorkflow can be tagged.
For this example,
            the SchemaMapping is tagged.
                """);
        try {
            actions.tagEntityResource(jsonSchemaMappingArn).join();
        } catch (CompletionException ce) {
            logger.error("Error tagging the resource: " +
ce.getCause().getMessage());
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. View the results of the AWS Entity Resolution Workflow.");
        logger.info("""
            You cannot view the result of the workflow that is in a running state.
            In order to view the results, you need to wait for the workflow that we
started in step 3 to complete.

            If you choose not to wait, you cannot view the results. You can perform

            this task manually in the AWS Management Console.

            This can take up to 30 mins (y/n).
            """);
        String viewAns = scanner.nextLine().trim();
        boolean isComplete = false;
        if (viewAns.equalsIgnoreCase("y")) {
            logger.info("You selected to view the Entity Resolution Workflow
results.");
            countdownWithWorkflowCheck(actions, 1800, jobId, workflowName);
            isComplete = true;
            try {
                JobMetrics metrics = actions.getJobInfo(workflowName, jobId).join();
                logger.info("Number of input records: {}", metrics.inputRecords());
                logger.info("Number of match ids: {}", metrics.matchIDs());
                logger.info("Number of records not processed: {}",
metrics.recordsNotProcessed());
                logger.info("Number of total records processed: {}",
metrics.totalRecordsProcessed());
```

```
                logger.info("The following represents the output data generated by
the Entity Resolution workflow based on the JSON and CSV input data. The output
data is stored in the {} bucket.", glueBucketName);
                actions.printData(glueBucketName);

                logger.info("""

                    Note that each of the last 2 records are considered a match even
though the 'name' differs between the records;
                    For example 'John Doe Jr.' compared to 'John Doe'.
                    The confidence level is a value between 0 and 1, where 1
indicates a perfect match.

                    """);

            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    logger.error("The job not found: {}", cause.getMessage(),
cause);
                } else {
                    logger.error("Error retrieving job information: " +
ce.getCause().getMessage());
                }
                return;
            }
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("9. Do you want to delete the resources, including the workflow?
(y/n)");
        logger.info("""
            You cannot delete the workflow that is in a running state.
            In order to delete the workflow, you need to wait for the workflow to
complete.

            You can delete the workflow manually in the AWS Management Console at a
later time.

            If you already waited for the workflow to complete in the previous
step,
```

```
            the workflow is completed and you can delete it.

            If the workflow is not completed, this can take up to 30 mins (y/n).
            """);
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            try {
                if (!isComplete) {
                    countdownWithWorkflowCheck(actions, 1800, jobId, workflowName);
                }
                actions.deleteMatchingWorkflowAsync(workflowName).join();
                logger.info("Workflow deleted successfully!");
            } catch (CompletionException ce) {
                logger.info("Error deleting the workflow: {} ", ce.getMessage());
                return;
            }

            try {
                // Delete both schema mappings.
                actions.deleteSchemaMappingAsync(jsonSchemaMappingName).join();
                actions.deleteSchemaMappingAsync(csvSchemaMappingName).join();
                logger.info("Both schema mappings were deleted successfully!");
            } catch (CompletionException ce) {
                logger.error("Error deleting schema mapping: {}", ce.getMessage());
                return;
            }

            waitForInputToContinue(scanner);
            logger.info(DASHES);
            logger.info("""
                Now we delete the CloudFormation stack, which deletes
                the resources that were created at the beginning of this scenario.
                """);
            waitForInputToContinue(scanner);
            logger.info(DASHES);
            try {
                deleteCloudFormationStack();
            } catch (RuntimeException e) {
                logger.error("Failed to delete the stack: {}", e.getMessage());
                return;
            }

        } else {
```

```
            logger.info("You can delete the AWS resources in the AWS Management
Console.");
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("This concludes the AWS Entity Resolution scenario.");
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }

    public static void countdownWithWorkflowCheck(EntityResActions actions, int
totalSeconds, String jobId, String workflowName) throws InterruptedException {
        int secondsElapsed = 0;

        while (true) {
            // Calculate display minutes and seconds.
            int remainingTime = totalSeconds - secondsElapsed;
            int displayMinutes = remainingTime / 60;
            int displaySeconds = remainingTime % 60;

            // Print the countdown.
            System.out.printf("\r%02d:%02d", displayMinutes, displaySeconds);
            Thread.sleep(1000); // Wait for 1 second
            secondsElapsed++;
```

```
            // Check workflow status every 60 seconds.
            if (secondsElapsed % 60 == 0 || remainingTime <= 0) {
                GetMatchingJobResponse response =
 actions.checkWorkflowStatusCompleteAsync(jobId, workflowName).join();
                if (response != null &&
 "SUCCEEDED".equalsIgnoreCase(String.valueOf(response.status()))) {
                    logger.info(""); // Move to the next line after countdown.
                    logger.info("Countdown complete: Workflow is in Completed
 state!");
                    break; // Break out of the loop if the status is "SUCCEEDED"
                }
            }

            // If countdown reaches zero, reset it for continuous countdown.
            if (remainingTime <= 0) {
                secondsElapsed = 0;
            }
        }
    }

    private static void deleteCloudFormationStack() {
        try {
            CloudFormationHelper.emptyS3Bucket(glueBucketName);
            CloudFormationHelper.destroyCloudFormationStack(STACK_NAME);
            logger.info("Resources deleted successfully!");
        } catch (CloudFormationException e) {
            throw new RuntimeException("Failed to delete CloudFormation stack: " +
 e.getMessage(), e);
        } catch (S3Exception e) {
            throw new RuntimeException("Failed to empty S3 bucket: " +
 e.getMessage(), e);
        }
    }
}
```

AWS Entity Resolution SDK 메서드의 래퍼 클래스입니다.

```
public class EntityResActions {

    private static final String PREFIX = "eroutput/";
    private static final Logger logger =
 LoggerFactory.getLogger(EntityResActions.class);
```

```
   private static EntityResolutionAsyncClient entityResolutionAsyncClient;

   private static S3AsyncClient s3AsyncClient;

   public static EntityResolutionAsyncClient getResolutionAsyncClient() {
       if (entityResolutionAsyncClient == null) {
           /*
           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.
               .retryStrategy(RetryMode.STANDARD)
               .build();

           entityResolutionAsyncClient = EntityResolutionAsyncClient.builder()
               .httpClient(httpClient)
               .overrideConfiguration(overrideConfig)
               .build();
       }
       return entityResolutionAsyncClient;
   }

   public static S3AsyncClient getS3AsyncClient() {
```

```
        if (s3AsyncClient == null) {
            /*
            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.
                .retryStrategy(RetryMode.STANDARD)
                .build();

            s3AsyncClient = S3AsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return s3AsyncClient;
    }

    /**
     * Deletes the schema mapping asynchronously.
     *
     * @param schemaName the name of the schema to delete
     * @return a {@link CompletableFuture} that completes when the schema mapping is
deleted successfully,
     * or throws a {@link RuntimeException} if the deletion fails
```

```
    */
   public CompletableFuture<DeleteSchemaMappingResponse>
deleteSchemaMappingAsync(String schemaName) {
       DeleteSchemaMappingRequest request = DeleteSchemaMappingRequest.builder()
           .schemaName(schemaName)
           .build();

       return getResolutionAsyncClient().deleteSchemaMapping(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   // Successfully deleted the schema mapping, log the success
message.
                   logger.info("Schema mapping '{}' deleted successfully.",
schemaName);
               } else {
                   // Ensure exception is not null before accessing its cause.
                   if (exception == null) {
                       throw new CompletionException("An unknown error occurred
while deleting the schema mapping.", null);
                   }

                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The schema mapping was not
found to delete: " + schemaName, cause);
                   }

                   // Wrap other AWS exceptions in a CompletionException.
                   throw new CompletionException("Failed to delete schema mapping:
" + schemaName, exception);
               }
           });
   }

   /**
    * Lists the schema mappings associated with the current AWS account. This
method uses an asynchronous paginator to
    * retrieve the schema mappings, and prints the name of each schema mapping to
the console.
    */
   public void ListSchemaMappings() {
       ListSchemaMappingsRequest mappingsRequest =
ListSchemaMappingsRequest.builder()
           .build();
```

```
        ListSchemaMappingsPublisher paginator =
getResolutionAsyncClient().listSchemaMappingsPaginator(mappingsRequest);

        // Iterate through the pages of results
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            response.schemaList().forEach(schemaMapping ->
                logger.info("Schema Mapping Name: " + schemaMapping.schemaName())
            );
        });

        // Wait for the asynchronous operation to complete
        future.join();
    }

    /**
     * Asynchronously deletes a workflow with the specified name.
     *
     * @param workflowName the name of the workflow to be deleted
     * @return a {@link CompletableFuture} that completes when the workflow has been
deleted
     * @throws RuntimeException if the deletion of the workflow fails
     */
    public CompletableFuture<DeleteMatchingWorkflowResponse>
deleteMatchingWorkflowAsync(String workflowName) {
        DeleteMatchingWorkflowRequest request =
DeleteMatchingWorkflowRequest.builder()
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().deleteMatchingWorkflow(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("{} was deleted", workflowName );
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while deleting the workflow.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The workflow to delete was
not found.", cause);
```

```
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to delete workflow: " +
exception.getMessage(), exception);
                }
            });
    }

    /**
     * Creates a schema mapping asynchronously.
     *
     * @param schemaName the name of the schema to create
     * @return a {@link CompletableFuture} that represents the asynchronous creation
of the schema mapping
     */
    public CompletableFuture<CreateSchemaMappingResponse>
createSchemaMappingAsync(String schemaName) {
        List<SchemaInputAttribute> schemaAttributes = null;
        if (schemaName.startsWith("json")) {
            schemaAttributes = List.of(

SchemaInputAttribute.builder().matchKey("id").fieldName("id").type(SchemaAttributeType.UNIQ

SchemaInputAttribute.builder().matchKey("name").fieldName("name").type(SchemaAttributeType.

SchemaInputAttribute.builder().matchKey("email").fieldName("email").type(SchemaAttributeTyp
            );
        } else {
            schemaAttributes = List.of(

SchemaInputAttribute.builder().matchKey("id").fieldName("id").type(SchemaAttributeType.UNIQ

SchemaInputAttribute.builder().matchKey("name").fieldName("name").type(SchemaAttributeType.

SchemaInputAttribute.builder().matchKey("email").fieldName("email").type(SchemaAttributeTyp

SchemaInputAttribute.builder().fieldName("phone").type(SchemaAttributeType.PROVIDER_ID).sub
            );
        }

        CreateSchemaMappingRequest request = CreateSchemaMappingRequest.builder()
            .schemaName(schemaName)
            .mappedInputFields(schemaAttributes)
```

```
            .build();

        return getResolutionAsyncClient().createSchemaMapping(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("[{}] schema mapping Created Successfully!",
schemaName);
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while creating the schema mapping.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflicting schema mapping
already exists. Resolve conflicts before proceeding.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to create schema mapping:
" + exception.getMessage(), exception);
                }
            });
    }

    /**
     * Retrieves the schema mapping asynchronously.
     *
     * @param schemaName the name of the schema to retrieve the mapping for
     * @return a {@link CompletableFuture} that completes with the {@link
GetSchemaMappingResponse} when the operation
     * is complete
     * @throws RuntimeException if the schema mapping retrieval fails
     */
    public CompletableFuture<GetSchemaMappingResponse> getSchemaMappingAsync(String
schemaName) {
        GetSchemaMappingRequest mappingRequest = GetSchemaMappingRequest.builder()
            .schemaName(schemaName)
            .build();

        return getResolutionAsyncClient().getSchemaMapping(mappingRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
```

```
                    response.mappedInputFields().forEach(attribute ->
                        logger.info("Attribute Name: " + attribute.fieldName() +
                            ", Attribute Type: " + attribute.type().toString()));
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while getting schema mapping.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested schema mapping
was not found.", cause);
                    }

                    // Wrap other exceptions in a CompletionException with the
message.
                    throw new CompletionException("Failed to get schema mapping: " +
exception.getMessage(), exception);
                }
            });
    }

    /**
     * Asynchronously retrieves a matching job based on the provided job ID and
workflow name.
     *
     * @param jobId        the ID of the job to retrieve
     * @param workflowName the name of the workflow associated with the job
     * @return a {@link CompletableFuture} that completes when the job information
is available or an exception occurs
     */
    public CompletableFuture<GetMatchingJobResponse> getMatchingJobAsync(String
jobId, String workflowName) {
        GetMatchingJobRequest request = GetMatchingJobRequest.builder()
            .jobId(jobId)
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().getMatchingJob(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    // Successfully fetched the matching job details, log the job
status.
```

```
                    logger.info("Job status: " + response.status());
                    logger.info("Job details: " + response.toString());
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while fetching the matching job.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested job could not
be found.", cause);
                    }

                    // Wrap other exceptions in a CompletionException with the
message.
                    throw new CompletionException("Error fetching matching job: " +
exception.getMessage(), exception);
                }
            });
    }


    /**
     * Starts a matching job asynchronously for the specified workflow name.
     *
     * @param workflowName the name of the workflow for which to start the matching
job
     * @return a {@link CompletableFuture} that completes with the job ID of the
started matching job, or an empty
     * string if the operation fails
     */
    public CompletableFuture<String> startMatchingJobAsync(String workflowName) {
        StartMatchingJobRequest jobRequest = StartMatchingJobRequest.builder()
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().startMatchingJob(jobRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    String jobId = response.jobId();
                    logger.info("Job ID: " + jobId);
                } else {
                    if (exception == null) {
```

```
                        throw new CompletionException("An unknown error occurred
while starting the job.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("The job is already running.
Resolve conflicts before starting a new job.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to start the job: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response != null ? response.jobId() : "");
    }

    /**
     * Checks the status of a workflow asynchronously.
     *
     * @param jobId        the ID of the job to check
     * @param workflowName the name of the workflow to check
     * @return a CompletableFuture that resolves to a boolean value indicating
whether the workflow has completed
     * successfully
     */
    public CompletableFuture<GetMatchingJobResponse>
checkWorkflowStatusCompleteAsync(String jobId, String workflowName) {
        GetMatchingJobRequest request = GetMatchingJobRequest.builder()
            .jobId(jobId)
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().getMatchingJob(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    // Process the response and log the job status.
                    logger.info("Job status: " + response.status());
                } else {
                    // Ensure exception is not null before accessing its cause.
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while checking job status.", null);
```

```
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested resource was
not found while checking the job status.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to check job status: " +
exception.getMessage(), exception);
                }
            });
    }

    /**
     * Creates an asynchronous CompletableFuture to manage the creation of a
matching workflow.
     *
     * @param roleARN                the AWS IAM role ARN to be used for the
workflow execution
     * @param workflowName           the name of the workflow to be created
     * @param outputBucket           the S3 bucket path where the workflow output
will be stored
     * @param jsonGlueTableArn       the ARN of the Glue Data Catalog table to be
used as the input source
     * @param jsonErSchemaMappingName the name of the schema to be used for the
input source
     * @return a CompletableFuture that, when completed, will return the ARN of the
created workflow
     */
    public CompletableFuture<String> createMatchingWorkflowAsync(
        String roleARN
        , String workflowName
        , String outputBucket
        , String jsonGlueTableArn
        , String jsonErSchemaMappingName
        , String csvGlueTableArn
        , String csvErSchemaMappingName) {

        InputSource jsonInputSource = InputSource.builder()
            .inputSourceARN(jsonGlueTableArn)
            .schemaName(jsonErSchemaMappingName)
            .applyNormalization(false)
```

```
            .build();

        InputSource csvInputSource = InputSource.builder()
            .inputSourceARN(csvGlueTableArn)
            .schemaName(csvErSchemaMappingName)
            .applyNormalization(false)
            .build();

        OutputAttribute idOutputAttribute = OutputAttribute.builder()
            .name("id")
            .build();

        OutputAttribute nameOutputAttribute = OutputAttribute.builder()
            .name("name")
            .build();

        OutputAttribute emailOutputAttribute = OutputAttribute.builder()
            .name("email")
            .build();

        OutputAttribute phoneOutputAttribute = OutputAttribute.builder()
            .name("phone")
            .build();

        OutputSource outputSource = OutputSource.builder()
            .outputS3Path("s3://" + outputBucket + "/eroutput")
            .output(idOutputAttribute, nameOutputAttribute, emailOutputAttribute,
phoneOutputAttribute)
            .applyNormalization(false)
            .build();

        ResolutionTechniques resolutionType = ResolutionTechniques.builder()
            .resolutionType(ResolutionType.ML_MATCHING)
            .build();

        CreateMatchingWorkflowRequest workflowRequest =
CreateMatchingWorkflowRequest.builder()
            .roleArn(roleARN)
            .description("Created by using the AWS SDK for Java")
            .workflowName(workflowName)
            .inputSourceConfig(List.of(jsonInputSource, csvInputSource))
            .outputSourceConfig(List.of(outputSource))
            .resolutionTechniques(resolutionType)
            .build();
```

```
        return getResolutionAsyncClient().createMatchingWorkflow(workflowRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Workflow created successfully.");
                } else {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ValidationException) {
                        throw new CompletionException("Invalid request: Please check
input parameters.", cause);
                    }

                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflicting workflow
already exists. Resolve conflicts before proceeding.", cause);
                    }
                    throw new CompletionException("Failed to create workflow: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(CreateMatchingWorkflowResponse::workflowArn);
    }

    /**
     * Tags the specified schema mapping ARN.
     *
     * @param schemaMappingARN the ARN of the schema mapping to tag
     */
    public CompletableFuture<TagResourceResponse> tagEntityResource(String
schemaMappingARN) {
        Map<String, String> tags = new HashMap<>();
        tags.put("tag1", "tag1Value");
        tags.put("tag2", "tag2Value");

        TagResourceRequest request = TagResourceRequest.builder()
            .resourceArn(schemaMappingARN)
            .tags(tags)
            .build();

        return getResolutionAsyncClient().tagResource(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    // Successfully tagged the resource, log the success message.
                    logger.info("Successfully tagged the resource.");
```

```
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while tagging the resource.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The resource to tag was not
found.", cause);
                    }
                    throw new CompletionException("Failed to tag the resource: " +
exception.getMessage(), exception);
                }
            });
    }

    public CompletableFuture<JobMetrics> getJobInfo(String workflowName, String
jobId) {
        return getResolutionAsyncClient().getMatchingJob(b -> b
                .workflowName(workflowName)
                .jobId(jobId))
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Job metrics fetched successfully for jobId: " +
jobId);
                } else {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("Invalid request: Job id was
not found.", cause);
                    }
                    throw new CompletionException("Failed to fetch job info: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.metrics()); // Extract job metrics
    }

    /**
     * Uploads data to an Amazon S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to upload the data to
     * @param jsonData   the JSON data to be uploaded
```

```
     * @param csvData    the CSV data to be uploaded
     * @return a {@link CompletableFuture} representing both asynchronous operation
of uploading the data
     * @throws RuntimeException if an error occurs during the file upload
     */

    public void uploadInputData(String bucketName, String jsonData, String csvData)
{
        // Upload JSON data.
        String jsonKey = "jsonData/data.json";
        PutObjectRequest jsonUploadRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(jsonKey)
            .contentType("application/json")
            .build();

        CompletableFuture<PutObjectResponse> jsonUploadResponse =
getS3AsyncClient().putObject(jsonUploadRequest,
AsyncRequestBody.fromString(jsonData));

        // Upload CSV data.
        String csvKey = "csvData/data.csv";
        PutObjectRequest csvUploadRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(csvKey)
            .contentType("text/csv")
            .build();
        CompletableFuture<PutObjectResponse> csvUploadResponse =
getS3AsyncClient().putObject(csvUploadRequest,
AsyncRequestBody.fromString(csvData));

        CompletableFuture.allOf(jsonUploadResponse, csvUploadResponse)
            .whenComplete((result, ex) -> {
                if (ex != null) {
                    // Wrap an AWS exception.
                    throw new CompletionException("Failed to upload files", ex);
                }
            }).join();

    }

    /**
     * Finds the latest file in the S3 bucket that starts with "run-" in any depth
of subfolders
```

```
     */
    private CompletableFuture<String> findLatestMatchingFile(String bucketName) {
        ListObjectsV2Request request = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .prefix(PREFIX) // Searches within the given folder
            .build();

        return getS3AsyncClient().listObjectsV2(request)
            .thenApply(response -> response.contents().stream()
                .map(S3Object::key)
                .filter(key -> key.matches(".*?/run-[0-9a-zA-Z\\-]+")) // Matches
files like run-XXXXX in any subfolder
                .max(String::compareTo) // Gets the latest file
                .orElse(null))
            .whenComplete((result, exception) -> {
                if (exception == null) {
                    if (result != null) {
                        logger.info("Latest matching file found: " + result);
                    } else {
                        logger.info("No matching files found.");
                    }
                } else {
                    throw new CompletionException("Failed to find latest matching
file: " + exception.getMessage(), exception);
                }
            });
    }

    /**
     * Prints the data located in the file in the S3 bucket that starts with "run-"
in any depth of subfolders
     */
    public void printData(String bucketName) {
        try {
            // Find the latest file with "run-" prefix in any depth of subfolders.
            String s3Key = findLatestMatchingFile(bucketName).join();
            if (s3Key == null) {
                logger.error("No matching files found in S3.");
                return;
            }

            logger.info("Downloading file: " + s3Key);

            // Read CSV file as String.
```

```
            String csvContent = readCSVFromS3Async(bucketName, s3Key).join();
            if (csvContent.isEmpty()) {
                logger.error("File is empty.");
                return;
            }

            // Process CSV content.
            List<String[]> records = parseCSV(csvContent);
            printTable(records);

        } catch (RuntimeException | IOException | CsvException e) {
            logger.error("Error processing CSV file from S3: " + e.getMessage());
            e.printStackTrace();
        }
    }

    /**
     * Reads a CSV file from S3 and returns it as a String.
     */
    private static CompletableFuture<String> readCSVFromS3Async(String bucketName,
String s3Key) {
        GetObjectRequest getObjectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(s3Key)
            .build();

        // Initiating the asynchronous request to get the file as bytes
        return getS3AsyncClient().getObject(getObjectRequest,
AsyncResponseTransformer.toBytes())
            .thenApply(responseBytes -> responseBytes.asUtf8String()) // Convert
bytes to UTF-8 string
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    throw new CompletionException("Failed to read CSV from S3: " +
exception.getMessage(), exception);
                } else {
                    logger.info("Successfully fetched CSV file content from S3.");
                }
            });
    }

    /**
     * Parses CSV content from a String into a list of records.
     */
```

```
    private static List<String[]> parseCSV(String csvContent) throws IOException,
CsvException {
        try (CSVReader csvReader = new CSVReader(new StringReader(csvContent))) {
            return csvReader.readAll();
        }
    }

    /**
     * Prints the given CSV data in a formatted table
     */
    private static void printTable(List<String[]> records) {
        if (records.isEmpty()) {
            System.out.println("No records found.");
            return;
        }

        String[] headers = records.get(0);
        List<String[]> rows = records.subList(1, records.size());

        // Determine column widths dynamically based on longest content
        int[] columnWidths = new int[headers.length];
        for (int i = 0; i < headers.length; i++) {
            final int columnIndex = i;
            int maxWidth = Math.max(headers[i].length(), rows.stream()
                .map(row -> row.length > columnIndex ? row[columnIndex].length() :
0)
                .max(Integer::compareTo)
                .orElse(0));
            columnWidths[i] = Math.min(maxWidth, 25); // Limit max width for better
readability
        }

        // Enable ANSI Console for colored output
        AnsiConsole.systemInstall();

        // Print table header
        System.out.println(ansi().fgYellow().a("=== CSV Data from S3 ===").reset());
        printRow(headers, columnWidths, true);

        // Print rows
        rows.forEach(row -> printRow(row, columnWidths, false));

        // Restore console to normal
        AnsiConsole.systemUninstall();
```

```
    }

    private static void printRow(String[] row, int[] columnWidths, boolean isHeader)
 {
        String border = IntStream.range(0, columnWidths.length)
            .mapToObj(i -> "-".repeat(columnWidths[i] + 2))
            .collect(Collectors.joining("+", "+", "+"));

        if (isHeader) {
            System.out.println(border);
        }

        System.out.print("|");
        for (int i = 0; i < columnWidths.length; i++) {
            String cell = (i < row.length && row[i] != null) ? row[i] : "";
            System.out.printf(" %-" + columnWidths[i] + "s |", isHeader ?
 ansi().fgBrightBlue().a(cell).reset() : cell);
        }
        System.out.println();

        if (isHeader) {
            System.out.println(border);
        }
    }
}
```

작업

`CreateMatchingWorkflow`

다음 코드 예시는 `CreateMatchingWorkflow`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Creates an asynchronous CompletableFuture to manage the creation of a
matching workflow.
    *
    * @param roleARN                the AWS IAM role ARN to be used for the
workflow execution
    * @param workflowName           the name of the workflow to be created
    * @param outputBucket           the S3 bucket path where the workflow output
will be stored
    * @param jsonGlueTableArn       the ARN of the Glue Data Catalog table to be
used as the input source
    * @param jsonErSchemaMappingName the name of the schema to be used for the
input source
    * @return a CompletableFuture that, when completed, will return the ARN of the
created workflow
    */
   public CompletableFuture<String> createMatchingWorkflowAsync(
       String roleARN
       , String workflowName
       , String outputBucket
       , String jsonGlueTableArn
       , String jsonErSchemaMappingName
       , String csvGlueTableArn
       , String csvErSchemaMappingName) {

       InputSource jsonInputSource = InputSource.builder()
           .inputSourceARN(jsonGlueTableArn)
           .schemaName(jsonErSchemaMappingName)
           .applyNormalization(false)
           .build();

       InputSource csvInputSource = InputSource.builder()
           .inputSourceARN(csvGlueTableArn)
           .schemaName(csvErSchemaMappingName)
           .applyNormalization(false)
           .build();

       OutputAttribute idOutputAttribute = OutputAttribute.builder()
           .name("id")
           .build();

       OutputAttribute nameOutputAttribute = OutputAttribute.builder()
           .name("name")
```

```
            .build();

        OutputAttribute emailOutputAttribute = OutputAttribute.builder()
            .name("email")
            .build();

        OutputAttribute phoneOutputAttribute = OutputAttribute.builder()
            .name("phone")
            .build();

        OutputSource outputSource = OutputSource.builder()
            .outputS3Path("s3://" + outputBucket + "/eroutput")
            .output(idOutputAttribute, nameOutputAttribute, emailOutputAttribute,
phoneOutputAttribute)
            .applyNormalization(false)
            .build();

        ResolutionTechniques resolutionType = ResolutionTechniques.builder()
            .resolutionType(ResolutionType.ML_MATCHING)
            .build();

        CreateMatchingWorkflowRequest workflowRequest =
CreateMatchingWorkflowRequest.builder()
            .roleArn(roleARN)
            .description("Created by using the AWS SDK for Java")
            .workflowName(workflowName)
            .inputSourceConfig(List.of(jsonInputSource, csvInputSource))
            .outputSourceConfig(List.of(outputSource))
            .resolutionTechniques(resolutionType)
            .build();

        return getResolutionAsyncClient().createMatchingWorkflow(workflowRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Workflow created successfully.");
                } else {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ValidationException) {
                        throw new CompletionException("Invalid request: Please check
input parameters.", cause);
                    }

                    if (cause instanceof ConflictException) {
```

```
                        throw new CompletionException("A conflicting workflow
already exists. Resolve conflicts before proceeding.", cause);
                    }
                    throw new CompletionException("Failed to create workflow: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(CreateMatchingWorkflowResponse::workflowArn);
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateMatchingWorkflow를 참조하세요.

`CreateSchemaMapping`

다음 코드 예시는 `CreateSchemaMapping`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Creates a schema mapping asynchronously.
     *
     * @param schemaName the name of the schema to create
     * @return a {@link CompletableFuture} that represents the asynchronous creation
of the schema mapping
     */
    public CompletableFuture<CreateSchemaMappingResponse>
createSchemaMappingAsync(String schemaName) {
        List<SchemaInputAttribute> schemaAttributes = null;
        if (schemaName.startsWith("json")) {
            schemaAttributes = List.of(

SchemaInputAttribute.builder().matchKey("id").fieldName("id").type(SchemaAttributeType.UNIQ

SchemaInputAttribute.builder().matchKey("name").fieldName("name").type(SchemaAttributeType.
```

```
SchemaInputAttribute.builder().matchKey("email").fieldName("email").type(SchemaAttributeTyp
            );
        } else {
            schemaAttributes = List.of(

SchemaInputAttribute.builder().matchKey("id").fieldName("id").type(SchemaAttributeType.UNIQ

SchemaInputAttribute.builder().matchKey("name").fieldName("name").type(SchemaAttributeType.

SchemaInputAttribute.builder().matchKey("email").fieldName("email").type(SchemaAttributeTyp

SchemaInputAttribute.builder().fieldName("phone").type(SchemaAttributeType.PROVIDER_ID).sub
            );
        }

        CreateSchemaMappingRequest request = CreateSchemaMappingRequest.builder()
            .schemaName(schemaName)
            .mappedInputFields(schemaAttributes)
            .build();

        return getResolutionAsyncClient().createSchemaMapping(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("[{}] schema mapping Created Successfully!",
schemaName);
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while creating the schema mapping.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflicting schema mapping
already exists. Resolve conflicts before proceeding.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to create schema mapping:
" + exception.getMessage(), exception);
                }
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateSchemaMapping을 참조하세요.

`DeleteMatchingWorkflow`

다음 코드 예시는 `DeleteMatchingWorkflow`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Asynchronously deletes a workflow with the specified name.
    *
    * @param workflowName the name of the workflow to be deleted
    * @return a {@link CompletableFuture} that completes when the workflow has been
deleted
    * @throws RuntimeException if the deletion of the workflow fails
    */
   public CompletableFuture<DeleteMatchingWorkflowResponse>
deleteMatchingWorkflowAsync(String workflowName) {
       DeleteMatchingWorkflowRequest request =
DeleteMatchingWorkflowRequest.builder()
           .workflowName(workflowName)
           .build();

       return getResolutionAsyncClient().deleteMatchingWorkflow(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   logger.info("{} was deleted", workflowName );
               } else {
                   if (exception == null) {
                       throw new CompletionException("An unknown error occurred
while deleting the workflow.", null);
                   }

                   Throwable cause = exception.getCause();
```

```
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The workflow to delete was
not found.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to delete workflow: " +
exception.getMessage(), exception);
                }
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteMatchingWorkflow를 참조하세요.

`DeleteSchemaMapping`

다음 코드 예시는 `DeleteSchemaMapping`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Deletes the schema mapping asynchronously.
     *
     * @param schemaName the name of the schema to delete
     * @return a {@link CompletableFuture} that completes when the schema mapping is
deleted successfully,
     * or throws a {@link RuntimeException} if the deletion fails
     */
    public CompletableFuture<DeleteSchemaMappingResponse>
deleteSchemaMappingAsync(String schemaName) {
        DeleteSchemaMappingRequest request = DeleteSchemaMappingRequest.builder()
            .schemaName(schemaName)
            .build();

        return getResolutionAsyncClient().deleteSchemaMapping(request)
```

```
            .whenComplete((response, exception) -> {
                if (response != null) {
                    // Successfully deleted the schema mapping, log the success
message.
                    logger.info("Schema mapping '{}' deleted successfully.",
schemaName);
                } else {
                    // Ensure exception is not null before accessing its cause.
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while deleting the schema mapping.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The schema mapping was not
found to delete: " + schemaName, cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to delete schema mapping:
" + schemaName, exception);
                }
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteSchemaMapping를 참조하세요.

`GetMatchingJob`

다음 코드 예시는 `GetMatchingJob`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
```

```
    * Asynchronously retrieves a matching job based on the provided job ID and
workflow name.
    *
    * @param jobId        the ID of the job to retrieve
    * @param workflowName the name of the workflow associated with the job
    * @return a {@link CompletableFuture} that completes when the job information
is available or an exception occurs
    */
   public CompletableFuture<GetMatchingJobResponse> getMatchingJobAsync(String
jobId, String workflowName) {
       GetMatchingJobRequest request = GetMatchingJobRequest.builder()
           .jobId(jobId)
           .workflowName(workflowName)
           .build();

       return getResolutionAsyncClient().getMatchingJob(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   // Successfully fetched the matching job details, log the job
status.
                   logger.info("Job status: " + response.status());
                   logger.info("Job details: " + response.toString());
               } else {
                   if (exception == null) {
                       throw new CompletionException("An unknown error occurred
while fetching the matching job.", null);
                   }

                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The requested job could not
be found.", cause);
                   }

                   // Wrap other exceptions in a CompletionException with the
message.
                   throw new CompletionException("Error fetching matching job: " +
exception.getMessage(), exception);
               }
           });
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetMatchingJob을 참조하세요.

GetSchemaMapping

다음 코드 예시는 GetSchemaMapping의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Retrieves the schema mapping asynchronously.
    *
    * @param schemaName the name of the schema to retrieve the mapping for
    * @return a {@link CompletableFuture} that completes with the {@link
GetSchemaMappingResponse} when the operation
    * is complete
    * @throws RuntimeException if the schema mapping retrieval fails
    */
   public CompletableFuture<GetSchemaMappingResponse> getSchemaMappingAsync(String
schemaName) {
       GetSchemaMappingRequest mappingRequest = GetSchemaMappingRequest.builder()
           .schemaName(schemaName)
           .build();

       return getResolutionAsyncClient().getSchemaMapping(mappingRequest)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   response.mappedInputFields().forEach(attribute ->
                       logger.info("Attribute Name: " + attribute.fieldName() +
                           ", Attribute Type: " + attribute.type().toString()));
               } else {
                   if (exception == null) {
                       throw new CompletionException("An unknown error occurred
while getting schema mapping.", null);
                   }

                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The requested schema mapping
was not found.", cause);
```

```
                    }

                    // Wrap other exceptions in a CompletionException with the
message.
                    throw new CompletionException("Failed to get schema mapping: " +
exception.getMessage(), exception);
                }
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetSchemaMapping을 참조하세요.

`ListSchemaMappings`

다음 코드 예시는 `ListSchemaMappings`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Lists the schema mappings associated with the current AWS account. This
method uses an asynchronous paginator to
     * retrieve the schema mappings, and prints the name of each schema mapping to
the console.
     */
    public void ListSchemaMappings() {
        ListSchemaMappingsRequest mappingsRequest =
ListSchemaMappingsRequest.builder()
            .build();

        ListSchemaMappingsPublisher paginator =
getResolutionAsyncClient().listSchemaMappingsPaginator(mappingsRequest);

        // Iterate through the pages of results
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            response.schemaList().forEach(schemaMapping ->
```

```
                logger.info("Schema Mapping Name: " + schemaMapping.schemaName())
            );
        });

        // Wait for the asynchronous operation to complete
        future.join();
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListSchemaMappings을 참조하세요.

`StartMatchingJob`

다음 코드 예시는 `StartMatchingJob`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Starts a matching job asynchronously for the specified workflow name.
     *
     * @param workflowName the name of the workflow for which to start the matching
job
     * @return a {@link CompletableFuture} that completes with the job ID of the
started matching job, or an empty
     * string if the operation fails
     */
    public CompletableFuture<String> startMatchingJobAsync(String workflowName) {
        StartMatchingJobRequest jobRequest = StartMatchingJobRequest.builder()
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().startMatchingJob(jobRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    String jobId = response.jobId();
```

```
                    logger.info("Job ID: " + jobId);
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while starting the job.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("The job is already running.
Resolve conflicts before starting a new job.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to start the job: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response != null ? response.jobId() : "");
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartMatchingJob을 참조하세요.

`TagResource`

다음 코드 예시는 `TagResource`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Tags the specified schema mapping ARN.
     *
     * @param schemaMappingARN the ARN of the schema mapping to tag
     */
```

```
   public CompletableFuture<TagResourceResponse> tagEntityResource(String
schemaMappingARN) {
       Map<String, String> tags = new HashMap<>();
       tags.put("tag1", "tag1Value");
       tags.put("tag2", "tag2Value");

       TagResourceRequest request = TagResourceRequest.builder()
           .resourceArn(schemaMappingARN)
           .tags(tags)
           .build();

       return getResolutionAsyncClient().tagResource(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   // Successfully tagged the resource, log the success message.
                   logger.info("Successfully tagged the resource.");
               } else {
                   if (exception == null) {
                       throw new CompletionException("An unknown error occurred
while tagging the resource.", null);
                   }

                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The resource to tag was not
found.", cause);
                   }
                   throw new CompletionException("Failed to tag the resource: " +
exception.getMessage(), exception);
               }
           });
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 TagResource를 참조하세요.

Java 2.x용 SDK를 사용하는 OpenSearch Service 예제

다음 코드 예제에서는 OpenSearch Service와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시작하기

OpenSearch Service 시작

다음 코드 예제에서는 OpenSearch Service 사용을 시작하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.services.opensearch.OpenSearchAsyncClient;
import software.amazon.awssdk.services.opensearch.model.ListVersionsRequest;
import java.util.List;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class HelloOpenSearch {
    public static void main(String[] args) {
        try {
            CompletableFuture<Void> future = listVersionsAsync();
            future.join();
            System.out.println("Versions listed successfully.");
        } catch (RuntimeException e) {
            System.err.println("Error occurred while listing versions: " +
 e.getMessage());
        }
    }

    private static OpenSearchAsyncClient getAsyncClient() {
        return OpenSearchAsyncClient.builder().build();
    }

    public static CompletableFuture<Void> listVersionsAsync() {
        ListVersionsRequest request = ListVersionsRequest.builder()
            .maxResults(10)
            .build();

        return getAsyncClient().listVersions(request).thenAccept(response -> {
            List<String> versionList = response.versions();
            for (String version : versionList) {
                System.out.println("Version info: " + version);
            }
        }).exceptionally(ex -> {
            // Handle the exception, or propagate it as a RuntimeException
            throw new RuntimeException("Failed to list versions", ex);
        });
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListVersions을 참조하세요.

기본 사항

OpenSearch Service 핵심 작업 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- OpenSearch Service 도메인을 생성합니다.

- 특정 OpenSearch Service 도메인에 대한 자세한 정보를 제공합니다.
- 계정이 소유한 모든 OpenSearch Service 도메인을 나열합니다.
- OpenSearch Service 도메인의 변경 상태가 완료 상태가 될 때까지 기다립니다.
- 기존 OpenSearch Service 도메인의 구성을 수정합니다.
- OpenSearch Service 도메인에 태그를 추가합니다.
- OpenSearch Service 도메인에 연결된 태그를 나열합니다.
- OpenSearch Service 도메인에서 태그를 제거합니다.
- OpenSearch Service 도메인을 삭제합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

OpenSearch Service 기능을 보여주는 대화형 시나리오를 실행합니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.opensearch.model.*;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;

public class OpenSearchScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    private static final Logger logger =
 LoggerFactory.getLogger(OpenSearchScenario.class);
    static Scanner scanner = new Scanner(System.in);

    static OpenSearchActions openSearchActions = new OpenSearchActions();

    public static void main(String[] args) throws Throwable {
        logger.info("""
            Welcome to the Amazon OpenSearch Service Basics Scenario.
```

```
            Use the Amazon OpenSearch Service API to create, configure, and manage
 OpenSearch Service domains.

            The operations exposed by the AWS OpenSearch Service client are focused
 on managing the OpenSearch Service domains
            and their configurations, not the data within the domains (such as
 indexing or querying documents).
            For document management, you typically interact directly with the
 OpenSearch REST API or use other libraries,
            such as the OpenSearch Java client (https://opensearch.org/docs/latest/
clients/java/).

            Let's get started...
        """);
        waitForInputToContinue(scanner);
        try {
            runScenario();
        } catch (RuntimeException e) {
            e.printStackTrace();
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }

    private static void runScenario() throws Throwable {
        String currentTimestamp = String.valueOf(System.currentTimeMillis());
        String domainName = "test-domain-" + currentTimestamp;

        logger.info(DASHES);
```

```
        logger.info("1. Create an Amazon OpenSearch domain");
        logger.info("""
            An Amazon OpenSearch domain is a managed instance of the OpenSearch
engine,
            which is an open-source search and analytics engine derived from
Elasticsearch.
            An OpenSearch domain is essentially a cluster of compute resources and
storage that hosts
            one or more OpenSearch indexes, enabling you to perform full-text
searches, data analysis, and
            visualizations.

            In this step, we'll initiate the creation of the domain. We'll check on
the progress in a later step.
        """);
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<String> future =
openSearchActions.createNewDomainAsync(domainName);
            String domainId = future.join();
            logger.info("Domain successfully created with ID: {}", domainId);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause != null) {
                if (cause instanceof OpenSearchException openSearchEx) {
                    logger.error("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
                } else {
                    logger.error("An unexpected error occurred: " +
cause.getMessage(), cause);
                }
            } else {
                logger.error("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("2. Describe the Amazon OpenSearch domain");
        logger.info("In this step, we get back the Domain ARN which is used in an
upcoming step.");
```

```
        waitForInputToContinue(scanner);

        String arn = "";
        try {
            CompletableFuture<String> future =
openSearchActions.describeDomainAsync(domainName);
            arn = future.join();
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("3. List the domains in your account");
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<List<DomainInfo>> future =
openSearchActions.listAllDomainsAsync();
            List<DomainInfo> domainInfoList = future.join();
            for (DomainInfo domain : domainInfoList) {
                logger.info("Domain name is: " + domain.domainName());
            }
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
OpenSearchException)) {
                cause = cause.getCause();
            }
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
```

```
            throw cause;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("4. Wait until the domain's change status reaches a completed
state");
        logger.info("""
            In this step, we check on the change status of the domain that we
initiated in Step 1.
            Until we reach a COMPLETED state, we stay in a loop by sending a
DescribeDomainChangeProgressRequest.

            The time it takes for a change to an OpenSearch domain to reach a
completed state can range
            from a few minutes to several hours. In this case the change is creating
a new domain that we initiated in Step 1.
            The time varies depending on the complexity of the change and the
current load on
            the OpenSearch service. In general, simple changes, such as scaling the
number of data nodes or
            updating the OpenSearch version, may take 10-30 minutes.
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<Void> future =
openSearchActions.domainChangeProgressAsync(domainName);
            future.join();
            logger.info("Domain change progress completed successfully.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
ResourceNotFoundException)) {
                cause = cause.getCause();
            }
            if (cause instanceof ResourceNotFoundException
resourceNotFoundException) {
                logger.info("The specific AWS resource was not found: Error message:
{}, Error code {}", resourceNotFoundException.awsErrorDetails().errorMessage(),
resourceNotFoundException.awsErrorDetails().errorCode());
```

```
                if (cause instanceof OpenSearchException ex) {
                    logger.info("An OpenSearch error occurred: Error message: " +
ex.getMessage());
                } else {
                    logger.info("An unexpected error occurred: " + rt.getMessage());
                }
                throw cause;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("5. Modify the domain");
        logger.info("""
            You can change your OpenSearch domain's settings, like the number of
instances, without starting over from scratch.
            This makes it easy to adjust your domain as your needs change, allowing
you to scale up or
            down quickly without recreating everything.

            We modify the domain in this step by changing the number of instances.
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<UpdateDomainConfigResponse> future =
openSearchActions.updateSpecificDomainAsync(domainName);
            UpdateDomainConfigResponse updateResponse = future.join();
            logger.info("Domain update status: " +
updateResponse.domainConfig().changeProgressDetails().configChangeStatusAsString());
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);
```

```
        logger.info("6. Wait until the domain's change status reaches a completed
state");
        logger.info("""
            In this step, we poll the status until the domain's change status
reaches a completed state.
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<Void> future =
openSearchActions.domainChangeProgressAsync(domainName);
            future.join();
            logger.info("Domain change progress completed successfully.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof OpenSearchException ex) {
                logger.info("EC2 error occurred: Error message: " +ex.getMessage());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("7. Tag the Domain");
        logger.info("""
            Tags let you assign arbitrary information to an Amazon OpenSearch
Service domain so you can
            categorize and filter on that information. A tag is a key-value pair
that you define and
            associate with an OpenSearch Service domain. You can use these tags to
track costs by grouping
            expenses for similarly tagged resources.

            In this scenario, we create tags with keys "service" and "instances".
        """);

        waitForInputToContinue(scanner);

        try {
```

```
            CompletableFuture<AddTagsResponse> future =
openSearchActions.addDomainTagsAsync(arn);
            future.join();
            logger.info("Domain tags added successfully.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
OpenSearchException)) {
                cause = cause.getCause();
            }
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
                if (cause != null) {
                    if (cause instanceof OpenSearchException) {
                        logger.error("OpenSearch error occurred: Error message: " +
cause.getMessage(), cause);
                    } else {
                        logger.error("An unexpected error occurred: " +
cause.getMessage(), cause);
                    }
                } else {
                    logger.error("An unexpected error occurred: " + rt.getMessage(),
rt);
                }
                throw cause;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("8. List Domain tags");
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<ListTagsResponse> future =
openSearchActions.listDomainTagsAsync(arn);
            ListTagsResponse listTagsResponse = future.join();
            listTagsResponse.tagList().forEach(tag -> logger.info("Tag Key: " +
tag.key() + ", Tag Value: " + tag.value()));
        } catch (RuntimeException rt) {
```

```
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
OpenSearchException)) {
                cause = cause.getCause();
            }
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;

        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("9. Delete the domain");
        logger.info("""
            In this step, we'll delete the Amazon OpenSearch domain that we created
in Step 1.
            Deleting a domain will remove all data and configuration for that
domain.
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<DeleteDomainResponse> future =
openSearchActions.deleteSpecificDomainAsync(domainName);
            future.join();
            logger.info("Domain successfully deleted.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
OpenSearchException)) {
                cause = cause.getCause();
            }
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
```

```
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;

        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("Scenario complete!");
    }
}
```

OpenSearch Service SDK 메서드의 래퍼 클래스입니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
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.opensearch.OpenSearchAsyncClient;
import software.amazon.awssdk.services.opensearch.model.AddTagsRequest;
import software.amazon.awssdk.services.opensearch.model.AddTagsResponse;
import software.amazon.awssdk.services.opensearch.model.ClusterConfig;
import software.amazon.awssdk.services.opensearch.model.CreateDomainRequest;
import software.amazon.awssdk.services.opensearch.model.DeleteDomainRequest;
import software.amazon.awssdk.services.opensearch.model.DeleteDomainResponse;
import
 software.amazon.awssdk.services.opensearch.model.DescribeDomainChangeProgressRequest;
import
 software.amazon.awssdk.services.opensearch.model.DescribeDomainChangeProgressResponse;
import software.amazon.awssdk.services.opensearch.model.DescribeDomainRequest;
import software.amazon.awssdk.services.opensearch.model.DomainInfo;
import software.amazon.awssdk.services.opensearch.model.DomainStatus;
import software.amazon.awssdk.services.opensearch.model.EBSOptions;
import software.amazon.awssdk.services.opensearch.model.ListDomainNamesRequest;
import software.amazon.awssdk.services.opensearch.model.ListTagsRequest;
import software.amazon.awssdk.services.opensearch.model.ListTagsResponse;
import software.amazon.awssdk.services.opensearch.model.NodeToNodeEncryptionOptions;
import software.amazon.awssdk.services.opensearch.model.Tag;
```

```
import software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigRequest;
import software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigResponse;
import software.amazon.awssdk.services.opensearch.model.VolumeType;
import java.time.Duration;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CompletableFuture;

public class OpenSearchActions {
    private static final Logger logger =
 LoggerFactory.getLogger(OpenSearchActions.class);
    private static OpenSearchAsyncClient openSearchClientAsyncClient;
    private static OpenSearchAsyncClient getAsyncClient() {
        if (openSearchClientAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryPolicy(RetryPolicy.builder()
                    .numRetries(3)
                    .build())
                .build();

            openSearchClientAsyncClient = OpenSearchAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return openSearchClientAsyncClient;
    }

    /**
     * Creates a new OpenSearch domain asynchronously.
     * @param domainName the name of the new OpenSearch domain to create
     * @return a {@link CompletableFuture} containing the domain ID of the newly
 created domain
```

```
     */
    public CompletableFuture<String> createNewDomainAsync(String domainName) {
        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();
        logger.info("Sending domain creation request...");
        return getAsyncClient().createDomain(domainRequest)
                .handle( (createResponse, throwable) -> {
                    if (createResponse != null) {
                        logger.info("Domain status is {}",
createResponse.domainStatus().changeProgressDetails().configChangeStatusAsString());
                        logger.info("Domain Id is {}",
createResponse.domainStatus().domainId());
                        return createResponse.domainStatus().domainId();
                    }
                    throw new RuntimeException("Failed to create domain",
throwable);
                });
    }

    /**
```

```
    * Deletes a specific domain asynchronously.
    * @param domainName the name of the domain to be deleted
    * @return a {@link CompletableFuture} that completes when the domain has been
deleted
    * or throws a {@link RuntimeException} if the deletion fails
    */
   public CompletableFuture<DeleteDomainResponse> deleteSpecificDomainAsync(String
domainName) {
       DeleteDomainRequest domainRequest = DeleteDomainRequest.builder()
           .domainName(domainName)
           .build();

       // Delete domain asynchronously
       return getAsyncClient().deleteDomain(domainRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   throw new RuntimeException("Failed to delete the domain: " +
domainName, exception);
               }
           });
   }

   /**
    * Describes the specified domain asynchronously.
    *
    * @param domainName the name of the domain to describe
    * @return a {@link CompletableFuture} that completes with the ARN of the domain
    * @throws RuntimeException if the domain description fails
    */
   public CompletableFuture<String> describeDomainAsync(String domainName) {
       DescribeDomainRequest request = DescribeDomainRequest.builder()
           .domainName(domainName)
           .build();

       return getAsyncClient().describeDomain(request)
           .handle((response, exception) -> {  // Handle both response and
exception
               if (exception != null) {
                   throw new RuntimeException("Failed to describe domain",
exception);
               }
               DomainStatus domainStatus = response.domainStatus();
               String endpoint = domainStatus.endpoint();
               String arn = domainStatus.arn();
```

```
                String engineVersion = domainStatus.engineVersion();
                logger.info("Domain endpoint is: " + endpoint);
                logger.info("ARN: " + arn);
                System.out.println("Engine version: " + engineVersion);

                return arn;  // Return ARN when successful
            });
    }

    /**
     * Asynchronously lists all the domains in the current AWS account.
     * @return a {@link CompletableFuture} that, when completed, contains a list of
{@link DomainInfo} objects representing
     *         the domains in the account.
     * @throws RuntimeException if there was a failure while listing the domains.
     */
    public CompletableFuture<List<DomainInfo>> listAllDomainsAsync() {
        ListDomainNamesRequest namesRequest = ListDomainNamesRequest.builder()
            .engineType("OpenSearch")
            .build();

        return getAsyncClient().listDomainNames(namesRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to list all domains",
exception);
                }
                return response.domainNames();  // Return the list of domain names
on success
            });
    }

    /**
     * Updates the configuration of a specific domain asynchronously.
     * @param domainName the name of the domain to update
     * @return a {@link CompletableFuture} that represents the asynchronous
operation of updating the domain configuration
     */
    public CompletableFuture<UpdateDomainConfigResponse>
updateSpecificDomainAsync(String domainName) {
        ClusterConfig clusterConfig = ClusterConfig.builder()
            .instanceCount(3)
            .build();
```

```
        UpdateDomainConfigRequest updateDomainConfigRequest =
UpdateDomainConfigRequest.builder()
            .domainName(domainName)
            .clusterConfig(clusterConfig)
            .build();

        return getAsyncClient().updateDomainConfig(updateDomainConfigRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to update the domain
configuration", exception);
                }
                // Handle success if needed (e.g., logging or additional actions)
            });
    }

    /**
     * Asynchronously checks the progress of a domain change operation in Amazon
OpenSearch Service.
     * @param domainName the name of the OpenSearch domain to check the progress for
     * @return a {@link CompletableFuture} that completes when the domain change
operation is completed
     */
    public CompletableFuture<Void> domainChangeProgressAsync(String domainName) {
        DescribeDomainChangeProgressRequest request =
DescribeDomainChangeProgressRequest.builder()
            .domainName(domainName)
            .build();

        return CompletableFuture.runAsync(() -> {
            boolean isCompleted = false;
            long startTime = System.currentTimeMillis();

            while (!isCompleted) {
                try {
                    // Handle the async client call using `join` to block
synchronously for the result
                    DescribeDomainChangeProgressResponse response = getAsyncClient()
                        .describeDomainChangeProgress(request)
                        .handle((resp, ex) -> {
                            if (ex != null) {
                                throw new RuntimeException("Failed to check domain
progress", ex);
                            }
```

```
                            return resp;
                        }).join();

                    String state = response.changeProgressStatus().statusAsString();
 // Get the status as string

                    if ("COMPLETED".equals(state)) {
                        logger.info("\nOpenSearch domain status: Completed");
                        isCompleted = true;
                    } else {
                        for (int i = 0; i < 5; i++) {
                            long elapsedTimeInSeconds = (System.currentTimeMillis()
- startTime) / 1000;
                            String formattedTime = String.format("%02d:%02d",
elapsedTimeInSeconds / 60, elapsedTimeInSeconds % 60);
                            System.out.print("\rOpenSearch domain state: " + state +
" | Time Elapsed: " + formattedTime + " ");
                            System.out.flush();
                            Thread.sleep(1_000);
                        }
                    }
                } catch (InterruptedException e) {
                    Thread.currentThread().interrupt();
                    throw new RuntimeException("Thread was interrupted", e);
                }
            }
        });
    }

    /**
     * Asynchronously adds tags to an Amazon OpenSearch Service domain.
     * @param domainARN the Amazon Resource Name (ARN) of the Amazon OpenSearch
Service domain to add tags to
     * @return a {@link CompletableFuture} that completes when the tags have been
successfully added to the domain,
     * or throws a {@link RuntimeException} if the operation fails
     */
    public CompletableFuture<AddTagsResponse> addDomainTagsAsync(String domainARN) {
        Tag tag1 = Tag.builder()
            .key("service")
            .value("OpenSearch")
            .build();

        Tag tag2 = Tag.builder()
```

```
            .key("instances")
            .value("m3.2xlarge")
            .build();

        List<Tag> tagList = new ArrayList<>();
        tagList.add(tag1);
        tagList.add(tag2);

        AddTagsRequest addTagsRequest = AddTagsRequest.builder()
            .arn(domainARN)
            .tagList(tagList)
            .build();

        return getAsyncClient().addTags(addTagsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to add tags to the domain: "
+ domainARN, exception);
                } else {
                    logger.info("Added Tags");
                }
            });
    }


    /**
     * Asynchronously lists the tags associated with the specified Amazon Resource
Name (ARN).
     * @param arn the Amazon Resource Name (ARN) of the resource for which to list
the tags
     * @return a {@link CompletableFuture} that, when completed, will contain a list
of the tags associated with the
     * specified ARN
     * @throws RuntimeException if there is an error listing the tags
     */
    public CompletableFuture<ListTagsResponse> listDomainTagsAsync(String arn) {
        ListTagsRequest tagsRequest = ListTagsRequest.builder()
            .arn(arn)
            .build();

        return getAsyncClient().listTags(tagsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
```

```
                    throw new RuntimeException("Failed to list domain tags",
 exception);
                }

                List<Tag> tagList = response.tagList();
                for (Tag tag : tagList) {
                    logger.info("Tag key is " + tag.key());
                    logger.info("Tag value is " + tag.value());
                }
            });
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - AddTags
  - CreateDomain
  - DeleteDomain
  - DescribeDomain
  - DescribeDomainChangeProgress
  - ListDomainNames
  - ListTags
  - UpdateDomainConfig

작업

`AddTags`

다음 코드 예시는 `AddTags`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Asynchronously adds tags to an Amazon OpenSearch Service domain.
     * @param domainARN the Amazon Resource Name (ARN) of the Amazon OpenSearch
Service domain to add tags to
     * @return a {@link CompletableFuture} that completes when the tags have been
successfully added to the domain,
     * or throws a {@link RuntimeException} if the operation fails
     */
    public CompletableFuture<AddTagsResponse> addDomainTagsAsync(String domainARN) {
        Tag tag1 = Tag.builder()
            .key("service")
            .value("OpenSearch")
            .build();

        Tag tag2 = Tag.builder()
            .key("instances")
            .value("m3.2xlarge")
            .build();

        List<Tag> tagList = new ArrayList<>();
        tagList.add(tag1);
        tagList.add(tag2);

        AddTagsRequest addTagsRequest = AddTagsRequest.builder()
            .arn(domainARN)
            .tagList(tagList)
            .build();

        return getAsyncClient().addTags(addTagsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to add tags to the domain: "
+ domainARN, exception);
                } else {
                    logger.info("Added Tags");
                }
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 AddTags를 참조하세요.

ChangeProgress

다음 코드 예시는 ChangeProgress의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Asynchronously checks the progress of a domain change operation in Amazon
OpenSearch Service.
    * @param domainName the name of the OpenSearch domain to check the progress for
    * @return a {@link CompletableFuture} that completes when the domain change
operation is completed
    */
   public CompletableFuture<Void> domainChangeProgressAsync(String domainName) {
       DescribeDomainChangeProgressRequest request =
DescribeDomainChangeProgressRequest.builder()
           .domainName(domainName)
           .build();

       return CompletableFuture.runAsync(() -> {
           boolean isCompleted = false;
           long startTime = System.currentTimeMillis();

           while (!isCompleted) {
               try {
                   // Handle the async client call using `join` to block
synchronously for the result
                   DescribeDomainChangeProgressResponse response = getAsyncClient()
                       .describeDomainChangeProgress(request)
                       .handle((resp, ex) -> {
                           if (ex != null) {
                               throw new RuntimeException("Failed to check domain
progress", ex);
                           }
                           return resp;
                       }).join();
```

```
                    String state = response.changeProgressStatus().statusAsString();
 // Get the status as string

                    if ("COMPLETED".equals(state)) {
                        logger.info("\nOpenSearch domain status: Completed");
                        isCompleted = true;
                    } else {
                        for (int i = 0; i < 5; i++) {
                            long elapsedTimeInSeconds = (System.currentTimeMillis()
- startTime) / 1000;
                            String formattedTime = String.format("%02d:%02d",
elapsedTimeInSeconds / 60, elapsedTimeInSeconds % 60);
                            System.out.print("\rOpenSearch domain state: " + state +
" | Time Elapsed: " + formattedTime + " ");
                            System.out.flush();
                            Thread.sleep(1_000);
                        }
                    }
                } catch (InterruptedException e) {
                    Thread.currentThread().interrupt();
                    throw new RuntimeException("Thread was interrupted", e);
                }
            }
        });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ChangeProgress를 참조하세요.

`CreateDomain`

다음 코드 예시는 `CreateDomain`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
```

```
    * Creates a new OpenSearch domain asynchronously.
    * @param domainName the name of the new OpenSearch domain to create
    * @return a {@link CompletableFuture} containing the domain ID of the newly
created domain
    */
   public CompletableFuture<String> createNewDomainAsync(String domainName) {
       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();
       logger.info("Sending domain creation request...");
       return getAsyncClient().createDomain(domainRequest)
               .handle( (createResponse, throwable) -> {
                   if (createResponse != null) {
                       logger.info("Domain status is {}",
createResponse.domainStatus().changeProgressDetails().configChangeStatusAsString());
                       logger.info("Domain Id is {}",
createResponse.domainStatus().domainId());
                       return createResponse.domainStatus().domainId();
                   }
                   throw new RuntimeException("Failed to create domain",
throwable);
```

```
                });
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateDomain을 참조하세요.

`DeleteDomain`

다음 코드 예시는 `DeleteDomain`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Deletes a specific domain asynchronously.
     * @param domainName the name of the domain to be deleted
     * @return a {@link CompletableFuture} that completes when the domain has been
deleted
     * or throws a {@link RuntimeException} if the deletion fails
     */
    public CompletableFuture<DeleteDomainResponse> deleteSpecificDomainAsync(String
domainName) {
        DeleteDomainRequest domainRequest = DeleteDomainRequest.builder()
            .domainName(domainName)
            .build();

        // Delete domain asynchronously
        return getAsyncClient().deleteDomain(domainRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to delete the domain: " +
domainName, exception);
                }
            });
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteDomain을 참조하세요.

`DescribeDomain`

다음 코드 예시는 `DescribeDomain`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Describes the specified domain asynchronously.
     *
     * @param domainName the name of the domain to describe
     * @return a {@link CompletableFuture} that completes with the ARN of the domain
     * @throws RuntimeException if the domain description fails
     */
    public CompletableFuture<String> describeDomainAsync(String domainName) {
        DescribeDomainRequest request = DescribeDomainRequest.builder()
            .domainName(domainName)
            .build();

        return getAsyncClient().describeDomain(request)
            .handle((response, exception) -> {  // Handle both response and
exception
                if (exception != null) {
                    throw new RuntimeException("Failed to describe domain",
exception);
                }
                DomainStatus domainStatus = response.domainStatus();
                String endpoint = domainStatus.endpoint();
                String arn = domainStatus.arn();
                String engineVersion = domainStatus.engineVersion();
                logger.info("Domain endpoint is: " + endpoint);
                logger.info("ARN: " + arn);
                System.out.println("Engine version: " + engineVersion);

                return arn;  // Return ARN when successful
```

```
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeDomain을 참조하세요.

`ListDomainNames`

다음 코드 예시는 `ListDomainNames`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Asynchronously lists all the domains in the current AWS account.
     * @return a {@link CompletableFuture} that, when completed, contains a list of
{@link DomainInfo} objects representing
     *         the domains in the account.
     * @throws RuntimeException if there was a failure while listing the domains.
     */
    public CompletableFuture<List<DomainInfo>> listAllDomainsAsync() {
        ListDomainNamesRequest namesRequest = ListDomainNamesRequest.builder()
            .engineType("OpenSearch")
            .build();

        return getAsyncClient().listDomainNames(namesRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to list all domains",
exception);
                }
                return response.domainNames();  // Return the list of domain names
on success
            });
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListDomainNames를 참조하세요.

`ListTags`

다음 코드 예시는 `ListTags`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Asynchronously adds tags to an Amazon OpenSearch Service domain.
    * @param domainARN the Amazon Resource Name (ARN) of the Amazon OpenSearch
Service domain to add tags to
    * @return a {@link CompletableFuture} that completes when the tags have been
successfully added to the domain,
    * or throws a {@link RuntimeException} if the operation fails
    */
   public CompletableFuture<AddTagsResponse> addDomainTagsAsync(String domainARN) {
       Tag tag1 = Tag.builder()
           .key("service")
           .value("OpenSearch")
           .build();

       Tag tag2 = Tag.builder()
           .key("instances")
           .value("m3.2xlarge")
           .build();

       List<Tag> tagList = new ArrayList<>();
       tagList.add(tag1);
       tagList.add(tag2);

       AddTagsRequest addTagsRequest = AddTagsRequest.builder()
           .arn(domainARN)
           .tagList(tagList)
           .build();
```

```
        return getAsyncClient().addTags(addTagsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to add tags to the domain: "
+ domainARN, exception);
                } else {
                    logger.info("Added Tags");
                }
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListTags를 참조하세요.

`UpdateDomainConfig`

다음 코드 예시는 `UpdateDomainConfig`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Updates the configuration of a specific domain asynchronously.
     * @param domainName the name of the domain to update
     * @return a {@link CompletableFuture} that represents the asynchronous
operation of updating the domain configuration
     */
    public CompletableFuture<UpdateDomainConfigResponse>
updateSpecificDomainAsync(String domainName) {
        ClusterConfig clusterConfig = ClusterConfig.builder()
            .instanceCount(3)
            .build();

        UpdateDomainConfigRequest updateDomainConfigRequest =
UpdateDomainConfigRequest.builder()
```

```
            .domainName(domainName)
            .clusterConfig(clusterConfig)
            .build();

        return getAsyncClient().updateDomainConfig(updateDomainConfigRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to update the domain
configuration", exception);
                }
                // Handle success if needed (e.g., logging or additional actions)
            });
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateDomainConfig을 참조하세요.

Java 2.x용 SDK를 사용하는 EventBridge 예제

다음 코드 예제에서는 EventBridge와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시작하기
- 기본 사항
- 작업
- 시나리오

시작하기

Hello EventBridge

다음 코드 예제에서는 EventBridge 사용을 시작하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * 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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListEventBuses를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 규칙을 만들고 여기에 대상을 추가하세요.
- 규칙을 활성화 및 비활성화합니다.
- 규칙과 대상을 나열하고 업데이트합니다.
- 이벤트를 전송하고 리소스를 정리합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including 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 "";
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - DeleteRule
  - DescribeRule
  - DisableRule
  - EnableRule
  - ListRuleNamesByTarget
  - ListRules
  - ListTargetsByRule
  - PutEvents
  - PutRule

- PutTargets

작업

`DeleteRule`

다음 코드 예시는 `DeleteRule`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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");
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteRule을 참조하세요.

`DescribeRule`

다음 코드 예시는 `DescribeRule`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeRule을 참조하세요.

`DisableRule`

다음 코드 예시는 `DisableRule`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

규칙 이름을 사용하여 규칙을 비활성화합니다.

```
   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);
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DisableRule을 참조하세요.

`EnableRule`

다음 코드 예시는 `EnableRule`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

규칙 이름을 사용하여 규칙을 활성화합니다.

```
    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);
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 EnableRule을 참조하세요.

`ListRuleNamesByTarget`

다음 코드 예시는 `ListRuleNamesByTarget`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

대상을 사용하여 모든 규칙 이름을 나열하세요.

```
   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);
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListRuleNamesByTarget을 참조하세요.

`ListRules`

다음 코드 예시는 `ListRules`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

규칙 이름을 사용하여 규칙을 활성화합니다.

```
    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);
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListRules를 참조하세요.

`ListTargetsByRule`

다음 코드 예시는 `ListTargetsByRule`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

규칙 이름을 사용하여 규칙의 모든 대상을 나열하세요.

```
    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());
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListTargetsByRule을 참조하세요.

`PutEvents`

다음 코드 예시는 `PutEvents`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutEvents를 참조하세요.

`PutRule`

다음 코드 예시는 `PutRule`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

예약된 규칙을 생성합니다.

```
   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);
       }
   }
```

객체가 Amazon Simple Storage Service 버킷에 추가될 때 트리거되는 규칙을 생성하세요.

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

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutRule을 참조하세요.

`PutTargets`

다음 코드 예시는 `PutTargets`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon SNS 주제를 규칙의 대상으로 추가합니다.

```
   // 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 + ".");
   }
```

규칙의 대상에 입력 트랜스포머를 추가합니다.

```
   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);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutTargets를 참조하세요.

`RemoveTargets`

다음 코드 예시는 `RemoveTargets`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

규칙 이름을 사용하여 규칙의 모든 대상을 제거합니다.

```
   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");
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 RemoveTargets를 참조하세요.

시나리오

EventBridge에 이벤트 알림 전송

다음 코드 예시에서는 S3 이벤트 알림을 EventBridge로 보내고 알림을 Amazon SNS 주제 및 Amazon SQS 대기열로 라우팅하도록 버킷을 설정하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /** This method configures a bucket to send events to AWS EventBridge and
creates a rule
     * to route the S3 object created events to a topic and a queue.
     *
     * @param bucketName Name of existing bucket
     * @param topicArn ARN of existing topic to receive S3 event notifications
     * @param queueArn ARN of existing queue to receive S3 event notifications
     *
     *  An AWS CloudFormation stack sets up the bucket, queue, topic before the
method runs.
     */
```

```
   public static String setBucketNotificationToEventBridge(String bucketName,
String topicArn, String queueArn) {
       try {
           // Enable bucket to emit S3 Event notifications to EventBridge.
           s3Client.putBucketNotificationConfiguration(b -> b
                   .bucket(bucketName)
                   .notificationConfiguration(b1 -> b1
                           .eventBridgeConfiguration(
                                   SdkBuilder::build)
                   ).build()).join();

           // Create an EventBridge rule to route Object Created notifications.
           PutRuleRequest putRuleRequest = PutRuleRequest.builder()
                   .name(RULE_NAME)
                   .eventPattern("""
                           {
                             "source": ["aws.s3"],
                             "detail-type": ["Object Created"],
                             "detail": {
                               "bucket": {
                                 "name": ["%s"]
                               }
                             }
                           }
                           """.formatted(bucketName))
                   .build();

           // Add the rule to the default event bus.
           PutRuleResponse putRuleResponse =
eventBridgeClient.putRule(putRuleRequest)
                   .whenComplete((r, t) -> {
                       if (t != null) {
                           logger.error("Error creating event bus rule: " +
t.getMessage(), t);
                           throw new RuntimeException(t.getCause().getMessage(),
t);
                       }
                       logger.info("Event bus rule creation request sent
successfully. ARN is: {}", r.ruleArn());
                   }).join();

           // Add the existing SNS topic and SQS queue as targets to the rule.
           eventBridgeClient.putTargets(b -> b
                   .eventBusName("default")
```

```
                .rule(RULE_NAME)
                .targets(List.of (
                        Target.builder()
                                .arn(queueArn)
                                .id("Queue")
                                .build(),
                        Target.builder()
                                .arn(topicArn)
                                .id("Topic")
                                .build())
                        )
                ).join();
            return putRuleResponse.ruleArn();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - PutBucketNotificationConfiguration
  - PutRule
  - PutTargets

예약된 이벤트를 사용하여 Lambda 함수 간접 호출

다음 코드 예제에서는 Amazon EventBridge 예약 이벤트에서 호출된 AWS Lambda 함수를 생성하는 방법을 보여줍니다.

SDK for Java 2.x

AWS Lambda 함수를 호출하는 Amazon EventBridge 예약 이벤트를 생성하는 방법을 보여줍니다. Lambda 함수가 간접 호출될 때 cron 표현식을 사용하여 일정을 예약하도록 EventBridge를 구성합니다. 이 예제에서는 Lambda Java 런타임 API를 사용하여 Lambda 함수를 생성합니다. 이 예제에서는 다양한 AWS 서비스를 호출하여 특정 사용 사례를 수행합니다. 이 예제에서는 1주년 기념일에 직원에게 축하하는 모바일 문자 메시지를 전송하는 앱을 생성하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

SDK for Java 2.x를 사용한 EventBridge Scheduler 예제

다음 코드 예제에서는 EventBridge 스케줄러와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시작하기

EventBridge Scheduler 시작

다음 코드 예제에서는 EventBridge 스케줄러 사용을 시작하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.services.scheduler.SchedulerAsyncClient;
import software.amazon.awssdk.services.scheduler.model.ListSchedulesRequest;
import software.amazon.awssdk.services.scheduler.model.ScheduleSummary;
import software.amazon.awssdk.services.scheduler.paginators.ListSchedulesPublisher;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CompletableFuture;

public class HelloScheduler {

    public static void main(String [] args) {
        listSchedulesAsync();
    }

    /**
     * Lists all the schedules available.
     * <p>
     * This method uses the {@link SchedulerAsyncClient} to make an asynchronous
 request to
     * list all the schedules available. The method uses the {@link
 ListSchedulesPublisher}
     * to fetch the schedules in a paginated manner, and then processes the
 responses
     * asynchronously.
     */
    public static void listSchedulesAsync() {
        SchedulerAsyncClient schedulerAsyncClient = SchedulerAsyncClient.create();

        // Build the request to list schedules
        ListSchedulesRequest listSchedulesRequest =
 ListSchedulesRequest.builder().build();

        // Use the paginator to fetch all schedules asynchronously.
```

```
        ListSchedulesPublisher paginator =
 schedulerAsyncClient.listSchedulesPaginator(listSchedulesRequest);
        List<ScheduleSummary> results = new ArrayList<>();

        // Subscribe to the paginator to process the response asynchronously
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            response.schedules().forEach(schedule -> {
                results.add(schedule);
                System.out.printf("Schedule: %s%n", schedule.name());
            });
        });

        // Wait for the asynchronous operation to complete.
        future.join();

        // After all schedules are fetched, print the total count.
        System.out.printf("Total of %d schedule(s) available.%n", results.size());
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListSchedules을 참조하세요.

작업

`CreateSchedule`

다음 코드 예시는 `CreateSchedule`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Creates a new schedule for a target task.
     *
```

```
    * @param name                   the name of the schedule
    * @param scheduleExpression    The schedule expression that defines when the
schedule should run.
    * @param scheduleGroupName     the name of the schedule group to which the
schedule belongs
    * @param targetArn             the Amazon Resource Name (ARN) of the target
task
    * @param roleArn               the ARN of the IAM role to be used for the
schedule
    * @param input                 the input data for the target task
    * @param deleteAfterCompletion whether to delete the schedule after it's
executed
    * @param useFlexibleTimeWindow whether to use a flexible time window for the
schedule execution
    * @return true if the schedule was successfully created, false otherwise
    */
   public CompletableFuture<Boolean> createScheduleAsync(
       String name,
       String scheduleExpression,
       String scheduleGroupName,
       String targetArn,
       String roleArn,
       String input,
       boolean deleteAfterCompletion,
       boolean useFlexibleTimeWindow) {

       int hoursToRun = 1;
       int flexibleTimeWindowMinutes = 10;

       Target target = Target.builder()
           .arn(targetArn)
           .roleArn(roleArn)
           .input(input)
           .build();

       FlexibleTimeWindow flexibleTimeWindow = FlexibleTimeWindow.builder()
           .mode(useFlexibleTimeWindow
               ? FlexibleTimeWindowMode.FLEXIBLE
               : FlexibleTimeWindowMode.OFF)
           .maximumWindowInMinutes(useFlexibleTimeWindow
               ? flexibleTimeWindowMinutes
               : null)
           .build();
```

```
        Instant startDate = Instant.now();
        Instant endDate = startDate.plus(Duration.ofHours(hoursToRun));

        CreateScheduleRequest request = CreateScheduleRequest.builder()
            .name(name)
            .scheduleExpression(scheduleExpression)
            .groupName(scheduleGroupName)
            .target(target)
            .actionAfterCompletion(deleteAfterCompletion
                ? ActionAfterCompletion.DELETE
                : ActionAfterCompletion.NONE)
            .startDate(startDate)
            .endDate(endDate)
            .flexibleTimeWindow(flexibleTimeWindow)
            .build();

        return getAsyncClient().createSchedule(request)
            .thenApply(response -> {
                logger.info("Successfully created schedule {} in schedule group {},
The ARN is {} ", name, scheduleGroupName, response.scheduleArn());
                return true;
            })
            .whenComplete((result, ex) -> {
                if (ex != null) {
                    if (ex instanceof ConflictException) {
                        // Handle ConflictException
                        logger.error("A conflict exception occurred while creating
the schedule: {}", ex.getMessage());
                        throw new CompletionException("A conflict exception occurred
while creating the schedule: " + ex.getMessage(), ex);
                    } else {
                        throw new CompletionException("Error creating schedule: " +
ex.getMessage(), ex);
                    }
                }
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateSchedule을 참조하세요.

`CreateScheduleGroup`

다음 코드 예시는 `CreateScheduleGroup`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Creates a new schedule group.
    *
    * @param name the name of the schedule group to be created
    * @return a {@link CompletableFuture} representing the asynchronous operation
of creating the schedule group
    */
   public CompletableFuture<CreateScheduleGroupResponse> createScheduleGroup(String
name) {
       CreateScheduleGroupRequest request = CreateScheduleGroupRequest.builder()
           .name(name)
           .build();

       logger.info("Initiating createScheduleGroup call for group: {}", name);
       CompletableFuture<CreateScheduleGroupResponse> futureResponse =
getAsyncClient().createScheduleGroup(request);
       futureResponse.whenComplete((response, ex) -> {
           if (ex != null) {
               if (ex instanceof CompletionException && ex.getCause() instanceof
ConflictException) {
                   // Rethrow the ConflictException
                   throw (ConflictException) ex.getCause();
               } else {
                   throw new CompletionException("Failed to create schedule group:
" + name, ex);
               }
           } else if (response == null) {
               throw new RuntimeException("Failed to create schedule group:
response was null");
           } else {
```

```
                logger.info("Successfully created schedule group '{}': {}", name,
response.scheduleGroupArn());
            }
        });

        return futureResponse;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateScheduleGroup을 참조하세요.

`DeleteSchedule`

다음 코드 예시는 `DeleteSchedule`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Deletes a schedule with the specified name and group name.
     *
     * @param name      the name of the schedule to be deleted
     * @param groupName the group name of the schedule to be deleted
     * @return a {@link CompletableFuture} that, when completed, indicates whether
the schedule was successfully deleted
     * @throws CompletionException if an error occurs while deleting the schedule,
except for the case where the schedule is not found
     */
    public CompletableFuture<Boolean> deleteScheduleAsync(String name, String
groupName) {
        DeleteScheduleRequest request = DeleteScheduleRequest.builder()
            .name(name)
            .groupName(groupName)
            .build();
```

```
        CompletableFuture<DeleteScheduleResponse> response =
getAsyncClient().deleteSchedule(request);
        return response.handle((result, ex) -> {
            if (ex != null) {
                if (ex instanceof ResourceNotFoundException) {
                    throw new CompletionException("Resource not found while deleting
schedule with ID: " + name, ex);
                } else {
                    throw new CompletionException("Failed to delete schedule.", ex);
                }
            }
            logger.info("Successfully deleted schedule with name {}.", name);
            return true;
        });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteSchedule을 참조하세요.

`DeleteScheduleGroup`

다음 코드 예시는 `DeleteScheduleGroup`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Deletes the specified schedule group.
     *
     * @param name the name of the schedule group to delete
     * @return a {@link CompletableFuture} that completes when the schedule group
has been deleted
     * @throws CompletionException if an error occurs while deleting the schedule
group
     */
```

```
    public CompletableFuture<Void> deleteScheduleGroupAsync(String name) {
        DeleteScheduleGroupRequest request = DeleteScheduleGroupRequest.builder()
            .name(name)
            .build();

        return getAsyncClient().deleteScheduleGroup(request)
            .thenRun(() -> {
                logger.info("Successfully deleted schedule group {}", name);
            })
            .whenComplete((result, ex) -> {
                if (ex != null) {
                    if (ex instanceof ResourceNotFoundException) {
                        throw new CompletionException("The resource was not found: "
+ ex.getMessage(), ex);
                    } else {
                        throw new CompletionException("Error deleting schedule
group: " + ex.getMessage(), ex);
                    }
                }
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteScheduleGroup을 참조하세요.

시나리오

예약된 이벤트

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 필요한 리소스가 포함된 CloudFormation 스택을 배포합니다.
- EventBridge Scheduler 일정 그룹을 만듭니다.
- 유연한 기간으로 일회성 EventBridge Scheduler 일정을 만듭니다.
- 지정된 속도로 반복 EventBridge Scheduler 일정을 만듭니다.
- EventBridge Scheduler 일정 및 일정 그룹을 삭제합니다.
- 리소스를 정리하고 스택을 삭제합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

시나리오를 실행합니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.scheduler.model.SchedulerException;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Map;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * This Java code example performs the following tasks for the Amazon EventBridge
 Scheduler workflow:
 * <p>
 * 1. Prepare the Application:
 * - Prompt the user for an email address to use for the subscription for the SNS
 topic subscription.
 * - Deploy the Cloud Formation template in resources/cfn_template.yaml for resource
 creation.
 * - Store the outputs of the stack into variables for use in the workflow.
 * - Create a schedule group for all workflow schedules.
 * <p>
 * 2. Create one-time Schedule:
 * - Create a one-time schedule to send an initial event.
 * - Use a Flexible Time Window and set the schedule to delete after completion.
 * - Wait for the user to receive the event email from SNS.
 * <p>
 * 3. Create a time-based schedule:
 * - Prompt the user for how many X times per Y hours a recurring event should be
 scheduled.
 * - Create the scheduled event for X times per hour for Y hours.
```

```
 * - Wait for the user to receive the event email from SNS.
 * - Delete the schedule when the user is finished.
 * <p>
 * 4. Clean up:
 * - Prompt the user for y/n answer if they want to destroy the stack and clean up
 all resources.
 * - Delete the schedule group.
 * - Destroy the Cloud Formation stack and wait until the stack has been removed.
 */

public class EventbridgeSchedulerScenario {

    private static final Logger logger =
 LoggerFactory.getLogger(EventbridgeSchedulerScenario.class);
    private static final Scanner scanner = new Scanner(System.in);
    private static String STACK_NAME = "workflow-stack-name";
    private static final String scheduleGroupName = "schedules-group";

    private static String recurringScheduleName = "";

    private static String oneTimeScheduleName = "";

    private static final EventbridgeSchedulerActions eventbridgeActions = new
 EventbridgeSchedulerActions();

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static String roleArn = "";
    public static String snsTopicArn = "";

    public static void main(String[] args) {
        logger.info(DASHES);
        logger.info("Welcome to the Amazon EventBridge Scheduler Workflow.");
        logger.info("""
            Amazon EventBridge Scheduler is a fully managed service that helps you
 schedule and execute
            a wide range of tasks and events in the cloud. It's designed to simplify
 the process of
            scheduling and managing recurring or one-time events, making it easier
 for developers and
            businesses to automate various workflows and processes.

            One of the key features of Amazon EventBridge Scheduler is its ability
 to schedule events
```

```
            based on a variety of triggers, including time-based schedules, custom
event patterns, or
            even integration with other AWS services. For example, you can use
EventBridge Scheduler
            to schedule a report generation task to run every weekday at 9 AM, or to
trigger a
            Lambda function when a specific Amazon S3 object is created.

            This flexibility allows you to build complex and dynamic event-driven
architectures
            that adapt to your business needs.

            Lets get started...
            """);
        waitForInputToContinue();
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("1. Prepare the application.");
        waitForInputToContinue();
        try {
            boolean prepareSuccess = prepareApplication();
            logger.info(DASHES);

            if (prepareSuccess) {
                logger.info("2. Create one-time schedule.");
                logger.info("""
                    A one-time schedule in Amazon EventBridge Scheduler is an event
trigger that allows
                    you to schedule a one-time event to run at a specific date and
time. This is useful for
                    executing a specific task or workflow at a predetermined time,
without the need for recurring
                    or complex scheduling.
                    """);
                waitForInputToContinue();
                createOneTimeSchedule();
                logger.info("Do you want to delete the schedule {} (y/n) ?",
oneTimeScheduleName);
                String ans = scanner.nextLine().trim();
                if (ans.equalsIgnoreCase("y")) {

eventbridgeActions.deleteScheduleAsync(oneTimeScheduleName,scheduleGroupName);
                }
```

```
                logger.info(DASHES);

                logger.info("3. Create a recurring schedule.");
                logger.info("""
                    A recurring schedule is a feature that allows you to schedule
and manage the execution
                    of your serverless applications or workloads on a recurring
basis. For example,
                    with EventBridge Scheduler, you can create custom schedules for
your AWS Lambda functions,
                    AWS Step Functions, and other supported event sources, enabling
you to automate tasks and
                    workflows without the need for complex infrastructure
management.
                    """);
                waitForInputToContinue();
                createRecurringSchedule();
                logger.info("Do you want to delete the schedule {} (y/n) ?",
oneTimeScheduleName);
                String ans2 = scanner.nextLine().trim();
                if (ans2.equalsIgnoreCase("y")) {

eventbridgeActions.deleteScheduleAsync(recurringScheduleName,scheduleGroupName);
                }
                logger.info(DASHES);
            }
        } catch (Exception ex) {
            logger.info("There was a problem with the workflow {}, initiating
cleanup...", ex.getMessage());
            cleanUp();
        }

        logger.info(DASHES);
        logger.info("4. Clean up the resources.");
        logger.info("Do you want to delete these AWS resources (y/n) ?");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            cleanUp();
        } else {
            logger.info("The AWS resources will not be deleted.");
        }
        logger.info("Amazon EventBridge Scheduler workflow completed.");
        logger.info(DASHES);
    }
```

```
   /**
    * Cleans up the resources associated with the EventBridge scheduler.
    * If any errors occur during the cleanup process, the corresponding error
messages are logged.
    */
   public static void cleanUp() {
       logger.info("First, delete the schedule group.");
       logger.info("When the schedule group is deleted, schedules that are part of
that group are deleted.");
       waitForInputToContinue();
       try {
           eventbridgeActions.deleteScheduleGroupAsync(scheduleGroupName).join();

       } catch (CompletionException ce) {
           Throwable cause = ce.getCause();
           if (cause instanceof SchedulerException schedulerException) {
               logger.error("Scheduler error occurred: Error message: {}, Error
code {}",
                   schedulerException.getMessage(),
schedulerException.awsErrorDetails().errorCode(), schedulerException);
           } else {
               logger.error("An unexpected error occurred: {}",
cause.getMessage());
           }
           return;
       }

       logger.info("Destroy the CloudFormation stack");
       waitForInputToContinue();
       CloudFormationHelper.destroyCloudFormationStack(STACK_NAME);
   }

   /**
    * Prepares the application by creating resources in a CloudFormation stack,
including an SNS topic
    * that will be subscribed to the EventBridge Scheduler events. The user will
need to confirm the subscription
    * in order to receive event emails.
    *
    * @return true if the application preparation was successful, false otherwise
    */
   public static boolean prepareApplication() {
       logger.info("""
```

```
            This example creates resources in a CloudFormation stack, including an
SNS topic
            that will be subscribed to the EventBridge Scheduler events.
            You will need to confirm the subscription in order to receive event
emails.
             """);

        String emailAddress = promptUserForEmail();
        logger.info("You entered {}", emailAddress);

        logger.info("Do you want to use a custom Stack name (y/n) ?");
        String ans = scanner.nextLine().trim();
        if (ans.equalsIgnoreCase("y")) {
            String newStackName = scanner.nextLine();
            logger.info("You entered {} for the new stack name", newStackName);
            waitForInputToContinue();
            STACK_NAME = newStackName;
        }

        logger.info("Get the roleArn and snsTopicArn values using a Cloudformation
template.");
        waitForInputToContinue();
        CloudFormationHelper.deployCloudFormationStack(STACK_NAME, emailAddress);
        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputs(STACK_NAME);
        roleArn = stackOutputs.get("RoleARN");
        snsTopicArn = stackOutputs.get("SNStopicARN");

        logger.info("The roleARN is {}", roleArn);
        logger.info("The snsTopicArn is {}", snsTopicArn);

        try {
            eventbridgeActions.createScheduleGroup(scheduleGroupName).join();
            logger.info("createScheduleGroupAsync completed successfully.");

        } catch (RuntimeException e) {
            logger.error("Error occurred: {} ", e.getMessage());
            return false;
        }
        logger.info("Application preparation complete.");
        return true;
    }

    /**
```

```
    * Waits for the user to enter 'c' followed by <ENTER> to continue the program.
    * This method is used to pause the program execution and wait for user input
before
    * proceeding.
    */
   private static void waitForInputToContinue() {
       while (true) {
           logger.info("");
           logger.info("Enter 'c' followed by <ENTER> to continue:");
           String input = scanner.nextLine();

           if (input.trim().equalsIgnoreCase("c")) {
               logger.info("Continuing with the program...");
               logger.info("");
               break;
           } else {
               // Handle invalid input.
               logger.info("Invalid input. Please try again.");
           }
       }
   }

   /**
    * Prompts the user to enter an email address and validates the input.
    * If the provided email address is invalid, the method will prompt the user to
try again.
    *
    * @return the valid email address entered by the user
    */
   private static String promptUserForEmail() {
       logger.info("Enter an email address to use for event subscriptions: ");
       String email = scanner.nextLine();
       if (!isValidEmail(email)) {
           logger.info("Invalid email address. Please try again.");
           return promptUserForEmail();
       }
       return email;
   }

   /**
    * Checks if the given email address is valid.
    *
    * @param email the email address to be validated
    * @return {@code true} if the email address is valid, {@code false} otherwise
```

```
     */
    private static boolean isValidEmail(String email) {
        try {
            InternetAddress emailAddress = new InternetAddress(email);
            emailAddress.validate();
            return true;

        } catch (AddressException e) {
            return false;
        }
    }

    /**
     * Creates a one-time schedule to send an initial event in 1 minute with a
 flexible time window.
     *
     * @return {@code true} if the schedule was created successfully, {@code false}
 otherwise
     */
    public static Boolean createOneTimeSchedule() {
        oneTimeScheduleName = promptUserForResourceName("Enter a name for the one-
time schedule:");
        logger.info("Creating a one-time schedule named {} to send an initial event
 in 1 minute with a flexible time window...", oneTimeScheduleName);
        LocalDateTime scheduledTime = LocalDateTime.now();
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-
dd'T'HH:mm:ss");

        String scheduleExpression = "at(" + scheduledTime.format(formatter) + ")";
        return eventbridgeActions.createScheduleAsync(
            oneTimeScheduleName,
            scheduleExpression,
            scheduleGroupName,
            snsTopicArn,
            roleArn,
            "One time scheduled event test from schedule",
            true,
            true).join();
    }


    /**
     * Creates a recurring schedule to send events based on a specific time.
     *
```

```
     * @return A {@link CompletableFuture} that completes with a boolean value
indicating the success or failure of the operation.
     */
    public static Boolean createRecurringSchedule() {
        logger.info("Creating a recurring schedule to send events for one hour...");
        recurringScheduleName = promptUserForResourceName("Enter a name for the
recurring schedule:");

        // Prompt the user for the schedule rate (in minutes).
        int scheduleRateInMinutes = promptUserForInteger("Enter the desired schedule
rate (in minutes): ");
        String scheduleExpression = "rate(" + scheduleRateInMinutes + " minutes)";
        return eventbridgeActions.createScheduleAsync(
            recurringScheduleName,
            scheduleExpression,
            scheduleGroupName,
            snsTopicArn,
            roleArn,
            "Recurrent event test from schedule " + recurringScheduleName,
            true,
            true).join();
    }

    /**
     * Prompts the user for a resource name and validates the input.
     *
     * @param prompt the message to display to the user when prompting for the
resource name
     * @return the valid resource name entered by the user
     */
    private static String promptUserForResourceName(String prompt) {
        logger.info(prompt);
        String resourceName = scanner.nextLine();
        String regex = "[0-9a-zA-Z-_.]+";
        if (!resourceName.matches(regex)) {
            logger.info("Invalid resource name. Please use a name that matches the
pattern " + regex + ".");
            return promptUserForResourceName(prompt);
        }
        return resourceName;
    }

    /**
     * Prompts the user for an integer input and returns the integer value.
```

```
     *
     * @param prompt the message to be displayed to the user when prompting for
 input
     * @return the integer value entered by the user
     */
    private static int promptUserForInteger(String prompt) {
        logger.info(prompt);
        String stringResponse = scanner.nextLine();
        if (stringResponse == null || stringResponse.trim().isEmpty() || !
isInteger(stringResponse)) {
            logger.info("Invalid integer.");
            return promptUserForInteger(prompt);
        }
        return Integer.parseInt(stringResponse);
    }

    /**
     * Checks if the given string represents a valid integer.
     *
     * @param str the string to be checked
     * @return {@code true} if the string represents a valid integer, {@code false}
 otherwise
     */
    private static boolean isInteger(String str) {
        try {
            Integer.parseInt(str);
            return true;
        } catch (NumberFormatException e) {
            return false;
        }
    }
}
```

서비스 작업용 래퍼입니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryMode;
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.scheduler.SchedulerAsyncClient;
import software.amazon.awssdk.services.scheduler.model.ActionAfterCompletion;
import software.amazon.awssdk.services.scheduler.model.ConflictException;
import software.amazon.awssdk.services.scheduler.model.CreateScheduleGroupRequest;
import software.amazon.awssdk.services.scheduler.model.CreateScheduleGroupResponse;
import software.amazon.awssdk.services.scheduler.model.CreateScheduleRequest;
import software.amazon.awssdk.services.scheduler.model.DeleteScheduleGroupRequest;
import software.amazon.awssdk.services.scheduler.model.DeleteScheduleRequest;
import software.amazon.awssdk.services.scheduler.model.DeleteScheduleResponse;
import software.amazon.awssdk.services.scheduler.model.FlexibleTimeWindow;
import software.amazon.awssdk.services.scheduler.model.FlexibleTimeWindowMode;
import software.amazon.awssdk.services.scheduler.model.ResourceNotFoundException;
import software.amazon.awssdk.services.scheduler.model.Target;

import java.time.Instant;
import java.util.concurrent.CompletableFuture;
import java.time.Duration;
import java.util.concurrent.CompletionException;

public class EventbridgeSchedulerActions {

    private static SchedulerAsyncClient schedulerClient;
    private static final Logger logger =
 LoggerFactory.getLogger(EventbridgeSchedulerActions.class);

    public static SchedulerAsyncClient getAsyncClient() {
        if (schedulerClient == null) {
            /*
            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.
                .retryStrategy(RetryMode.STANDARD)
                .build();

            schedulerClient = SchedulerAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return schedulerClient;
    }


    /**
     * Creates a new schedule group.
     *
     * @param name the name of the schedule group to be created
     * @return a {@link CompletableFuture} representing the asynchronous operation
of creating the schedule group
     */
    public CompletableFuture<CreateScheduleGroupResponse> createScheduleGroup(String
name) {
        CreateScheduleGroupRequest request = CreateScheduleGroupRequest.builder()
            .name(name)
            .build();

        logger.info("Initiating createScheduleGroup call for group: {}", name);
        CompletableFuture<CreateScheduleGroupResponse> futureResponse =
getAsyncClient().createScheduleGroup(request);
        futureResponse.whenComplete((response, ex) -> {
            if (ex != null) {
                if (ex instanceof CompletionException && ex.getCause() instanceof
ConflictException) {
                    // Rethrow the ConflictException
                    throw (ConflictException) ex.getCause();
```

```
                } else {
                    throw new CompletionException("Failed to create schedule group:
" + name, ex);
                }
            } else if (response == null) {
                throw new RuntimeException("Failed to create schedule group:
response was null");
            } else {
                logger.info("Successfully created schedule group '{}': {}", name,
response.scheduleGroupArn());
            }
        });

        return futureResponse;
    }


    /**
     * Creates a new schedule for a target task.
     *
     * @param name                  the name of the schedule
     * @param scheduleExpression    The schedule expression that defines when the
schedule should run.
     * @param scheduleGroupName     the name of the schedule group to which the
schedule belongs
     * @param targetArn             the Amazon Resource Name (ARN) of the target
task
     * @param roleArn               the ARN of the IAM role to be used for the
schedule
     * @param input                 the input data for the target task
     * @param deleteAfterCompletion whether to delete the schedule after it's
executed
     * @param useFlexibleTimeWindow whether to use a flexible time window for the
schedule execution
     * @return true if the schedule was successfully created, false otherwise
     */
    public CompletableFuture<Boolean> createScheduleAsync(
        String name,
        String scheduleExpression,
        String scheduleGroupName,
        String targetArn,
        String roleArn,
        String input,
        boolean deleteAfterCompletion,
```

```
        boolean useFlexibleTimeWindow) {

        int hoursToRun = 1;
        int flexibleTimeWindowMinutes = 10;

        Target target = Target.builder()
            .arn(targetArn)
            .roleArn(roleArn)
            .input(input)
            .build();

        FlexibleTimeWindow flexibleTimeWindow = FlexibleTimeWindow.builder()
            .mode(useFlexibleTimeWindow
                ? FlexibleTimeWindowMode.FLEXIBLE
                : FlexibleTimeWindowMode.OFF)
            .maximumWindowInMinutes(useFlexibleTimeWindow
                ? flexibleTimeWindowMinutes
                : null)
            .build();

        Instant startDate = Instant.now();
        Instant endDate = startDate.plus(Duration.ofHours(hoursToRun));

        CreateScheduleRequest request = CreateScheduleRequest.builder()
            .name(name)
            .scheduleExpression(scheduleExpression)
            .groupName(scheduleGroupName)
            .target(target)
            .actionAfterCompletion(deleteAfterCompletion
                ? ActionAfterCompletion.DELETE
                : ActionAfterCompletion.NONE)
            .startDate(startDate)
            .endDate(endDate)
            .flexibleTimeWindow(flexibleTimeWindow)
            .build();

        return getAsyncClient().createSchedule(request)
            .thenApply(response -> {
                logger.info("Successfully created schedule {} in schedule group {},
The ARN is {} ", name, scheduleGroupName, response.scheduleArn());
                return true;
            })
            .whenComplete((result, ex) -> {
                if (ex != null) {
```

```
                    if (ex instanceof ConflictException) {
                        // Handle ConflictException
                        logger.error("A conflict exception occurred while creating
the schedule: {}", ex.getMessage());
                        throw new CompletionException("A conflict exception occurred
while creating the schedule: " + ex.getMessage(), ex);
                    } else {
                        throw new CompletionException("Error creating schedule: " +
ex.getMessage(), ex);
                    }
                }
            });
    }


    /**
     * Deletes the specified schedule group.
     *
     * @param name the name of the schedule group to delete
     * @return a {@link CompletableFuture} that completes when the schedule group
has been deleted
     * @throws CompletionException if an error occurs while deleting the schedule
group
     */
    public CompletableFuture<Void> deleteScheduleGroupAsync(String name) {
        DeleteScheduleGroupRequest request = DeleteScheduleGroupRequest.builder()
            .name(name)
            .build();

        return getAsyncClient().deleteScheduleGroup(request)
            .thenRun(() -> {
                logger.info("Successfully deleted schedule group {}", name);
            })
            .whenComplete((result, ex) -> {
                if (ex != null) {
                    if (ex instanceof ResourceNotFoundException) {
                        throw new CompletionException("The resource was not found: "
+ ex.getMessage(), ex);
                    } else {
                        throw new CompletionException("Error deleting schedule
group: " + ex.getMessage(), ex);
                    }
                }
            });
```

```
    }


    /**
     * Deletes a schedule with the specified name and group name.
     *
     * @param name      the name of the schedule to be deleted
     * @param groupName the group name of the schedule to be deleted
     * @return a {@link CompletableFuture} that, when completed, indicates whether
 the schedule was successfully deleted
     * @throws CompletionException if an error occurs while deleting the schedule,
 except for the case where the schedule is not found
     */
    public CompletableFuture<Boolean> deleteScheduleAsync(String name, String
 groupName) {
        DeleteScheduleRequest request = DeleteScheduleRequest.builder()
            .name(name)
            .groupName(groupName)
            .build();

        CompletableFuture<DeleteScheduleResponse> response =
 getAsyncClient().deleteSchedule(request);
        return response.handle((result, ex) -> {
            if (ex != null) {
                if (ex instanceof ResourceNotFoundException) {
                    throw new CompletionException("Resource not found while deleting
 schedule with ID: " + name, ex);
                } else {
                    throw new CompletionException("Failed to delete schedule.", ex);
                }
            }
            logger.info("Successfully deleted schedule with name {}.", name);
            return true;
        });
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - CreateSchedule
  - CreateScheduleGroup
  - DeleteSchedule

- DeleteScheduleGroups

Java 2.x용 SDK를 사용하는 Forecast 예제

다음 코드 예제에서는 Forecast와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`CreateDataset`

다음 코드 예시는 `CreateDataset`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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;
    }
```

```
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateDataset를 참조하세요.

`CreateForecast`

다음 코드 예시는 `CreateForecast`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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 "";
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateForecast를 참조하세요.

DeleteDataset

다음 코드 예시는 DeleteDataset의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteDataset를 참조하세요.

`DeleteForecast`

다음 코드 예시는 `DeleteForecast`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteForecast를 참조하세요.

`DescribeForecast`

다음 코드 예시는 `DescribeForecast`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeForecast를 참조하세요.

`ListDatasetGroups`

다음 코드 예시는 `ListDatasetGroups`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListDatasetGroups를 참조하세요.

`ListForecasts`

다음 코드 예시는 `ListForecasts`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListForecasts를 참조하세요.

SDK for Java 2.x를 사용하는 Amazon Glacier 예제

다음 코드 예제에서는 AWS SDK for Java 2.x Amazon Glacier에서를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`CreateVault`

다음 코드 예시는 `CreateVault`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
```

```
}
```

- API 세부 정보는AWS SDK for Java 2.x API 참조의 CreateVault를 참조하세요.

`DeleteArchive`

다음 코드 예시는 `DeleteArchive`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteArchive를 참조하세요.

`DeleteVault`

다음 코드 예시는 `DeleteVault`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteVault를 참조하세요.

`InitiateJob`

다음 코드 예시는 `InitiateJob`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

저장소 인벤토리를 가져옵니다.

```
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);

        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 InitiateJob을 참조하세요.

`ListVaults`

다음 코드 예시는 `ListVaults`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListVaults를 참조하세요.

`UploadArchive`

다음 코드 예시는 `UploadArchive`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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();
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 UploadArchive를 참조하세요.

AWS Glue SDK for Java 2.x를 사용한 예제

다음 코드 예제에서는와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS Glue.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시작하기

안녕하세요 AWS Glue

다음 코드 예제에서는 AWS Glue를 사용하여 시작하는 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        });
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListJobs를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 퍼블릭 Amazon S3 버킷을 크롤링하고 CSV 형식의 메타데이터 데이터베이스를 생성하는 크롤러를 생성합니다.
- 의 데이터베이스 및 테이블에 대한 정보를 나열합니다 AWS Glue Data Catalog.
- 작업을 생성하여 S3 버킷에서 CSV 데이터를 추출하고, 데이터를 변환하며, JSON 형식의 출력을 다른 S3 버킷으로 로드합니다.
- 작업 실행에 대한 정보를 나열하고 변환된 데이터를 확인하며 리소스를 정리합니다.

자세한 내용은 자습서: AWS Glue Studio 시작하기를 참조하세요.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * 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>
 <scriptLocation> <locationUri> <bucketNameSc>\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, s3://<bucket name>/read).
                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 (you can find this file
 in resources folder).
                bucketNameSc - The Amazon S3 bucket name used when creating a job
                """;

        if (args.length != 9) {
            System.out.println(usage);
            return;
        }
        Scanner scanner = new Scanner(System.in);
        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("""
```

```
            AWS Glue is a fully managed extract, transform, and load (ETL) service
provided by Amazon
            Web Services (AWS). It is designed to simplify the process of building,
running, and maintaining
            ETL pipelines, which are essential for data integration and data
warehousing tasks.

            One of the key features of AWS Glue is its ability to automatically
discover and catalog data
            stored in various sources, such as Amazon S3, Amazon RDS, Amazon
Redshift, and other databases.
            This cataloging process creates a central metadata repository, known as
the AWS Glue Data Catalog,
            which provides a unified view of an organization's data assets. This
metadata can then be used to
            create ETL jobs, which can be scheduled and run on-demand or on a
regular basis.

            Lets get started.

            """);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create a database.");
        try {
            createDatabase(glueClient, dbName, locationUri);
        } catch (GlueException e) {
            if (e.awsErrorDetails().errorMessage().equals("Database already
exists.")) {
                System.out.println("Database " + dbName + " already exists. Skipping
creation.");
            } else {
                System.err.println(e.awsErrorDetails().errorMessage());
                return;
            }
        }

        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a crawler.");
```

```
        try {
            createGlueCrawler(glueClient, iam, s3Path, cron, dbName, crawlerName);
        } catch (GlueException e) {
            if (e.awsErrorDetails().errorMessage().contains("already exists")) {
                System.out.println("Crawler " + crawlerName + " already exists.
Skipping creation.");
            } else {
                System.err.println(e.awsErrorDetails().errorMessage());
                System.exit(1);
            }
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get a crawler.");
        try {
            getSpecificCrawler(glueClient, crawlerName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Start a crawler.");
        try {
            startSpecificCrawler(glueClient, crawlerName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Get a database.");
        try {
            getSpecificDatabase(glueClient, dbName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
```

```
        waitForInputToContinue(scanner);
        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;
        try {
            myTableName = getGlueTables(glueClient, dbName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Create a job.");
        try {
            createJob(glueClient, jobName, iam, scriptLocation);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Start a Job run.");
        try {
            startJob(glueClient, jobName, dbName, myTableName, bucketNameSc);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. List all jobs.");
        try {
            getAllJobs(glueClient);
        } catch (GlueException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get job runs.");
        try {
            getJobRuns(glueClient, jobName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete a job.");
        try {
            deleteJob(glueClient, jobName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        System.out.println("*** Wait 5 MIN for the " + crawlerName + " to stop");
        TimeUnit.MINUTES.sleep(5);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Delete a database.");
        try {
            deleteDatabase(glueClient, dbName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Delete a crawler.");
        try {
```

```
            deleteSpecificCrawler(glueClient, crawlerName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Successfully completed the AWS Glue Scenario");
        System.out.println(DASHES);
    }


    /**
     * Creates a Glue database with the specified name and location URI.
     *
     * @param glueClient  The Glue client to use for the database creation.
     * @param dbName      The name of the database to create.
     * @param locationUri The location URI for the database.
     */
    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) {
            throw e;
        }
    }


    /**
```

```
    * Creates a new AWS Glue crawler using the AWS Glue Java API.
    *
    * @param glueClient  the AWS Glue client used to interact with the AWS Glue
service
    * @param iam         the IAM role that the crawler will use to access the data
source
    * @param s3Path      the S3 path that the crawler will scan for data
    * @param cron        the cron expression that defines the crawler's schedule
    * @param dbName      the name of the AWS Glue database where the crawler will
store the metadata
    * @param crawlerName the name of the crawler to be created
    */
   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) {
           throw e;
```

```
        }
    }

    /**
     * Retrieves a specific crawler from the AWS Glue service and waits for it to be
in the "READY" state.
     *
     * @param glueClient  the AWS Glue client used to interact with the Glue service
     * @param crawlerName the name of the crawler to be retrieved
     */
    public static void getSpecificCrawler(GlueClient glueClient, String crawlerName)
throws InterruptedException {
        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) {
            throw e;
        }
    }

    /**
     * Starts a specific AWS Glue crawler.
     *
     * @param glueClient  the AWS Glue client to use for the crawler operation
     * @param crawlerName the name of the crawler to start
     * @throws GlueException if there is an error starting the crawler
     */
    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) {
            throw e;
        }
    }

    /**
     * Retrieves the specific database from the AWS Glue service.
     *
     * @param glueClient   an instance of the AWS Glue client used to interact with
the service
     * @param databaseName the name of the database to retrieve
     * @throws GlueException if there is an error retrieving the database from the
AWS Glue service
     */
    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) {
            throw e;
        }
    }
```

```
   /**
    * Retrieves the names of the tables in the specified Glue database.
    *
    * @param glueClient the Glue client to use for the operation
    * @param dbName     the name of the Glue database to retrieve the table names
from
    * @return the name of the first table retrieved, or an empty string if no
tables were found
    */
   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) {
           throw e;
       }
       return myTableName;
   }


   /**
    * Starts a job run in AWS Glue.
    *
    * @param glueClient    the AWS Glue client to use for the job run
    * @param jobName       the name of the Glue job to run
    * @param inputDatabase the name of the input database
    * @param inputTable    the name of the input table
    * @param outBucket     the URL of the output S3 bucket
```

```
     * @throws GlueException if there is an error starting the job run
     */
    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) {
            throw e;
        }
    }


    /**
     * Creates a new AWS Glue job.
     *
     * @param glueClient     the AWS Glue client to use for the operation
     * @param jobName        the name of the job to create
     * @param iam            the IAM role to associate with the job
     * @param scriptLocation the location of the script to be used by the job
     * @throws GlueException if there is an error creating the job
     */
    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) {
            throw e;
        }
    }


    /**
     * Retrieves and prints information about all the jobs in the Glue data catalog.
     *
     * @param glueClient the Glue client used to interact with the AWS Glue service
     */
    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) {
            throw e;
        }
    }
```

```
   /**
    * Retrieves the job runs for a given Glue job and prints the status of the job
runs.
    *
    * @param glueClient the Glue client used to make API calls
    * @param jobName    the name of the Glue job to retrieve the job runs for
    */
   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 e) {
            throw e;
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }


    /**
     * Deletes a Glue job.
     *
     * @param glueClient the Glue client to use for the operation
     * @param jobName    the name of the job to be deleted
     * @throws GlueException if there is an error deleting the job
     */
    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) {
            throw e;
        }
    }

    /**
     * Deletes a AWS Glue Database.
     *
     * @param glueClient   An instance of the AWS Glue client used to interact with
the AWS Glue service.
     * @param databaseName The name of the database to be deleted.
     * @throws GlueException If an error occurs while deleting the database.
     */
    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) {
            throw e;
        }
    }


    /**
     * Deletes a specific AWS Glue crawler.
     *
     * @param glueClient  the AWS Glue client object
     * @param crawlerName the name of the crawler to be deleted
     * @throws GlueException if an error occurs during the deletion process
     */
    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) {
            throw e;
        }
    }

    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.");
            }
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하십시오.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

작업

`CreateCrawler`

다음 코드 예시는 `CreateCrawler`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Creates a new AWS Glue crawler using the AWS Glue Java API.
    *
    * @param glueClient  the AWS Glue client used to interact with the AWS Glue
service
    * @param iam         the IAM role that the crawler will use to access the data
source
    * @param s3Path      the S3 path that the crawler will scan for data
    * @param cron        the cron expression that defines the crawler's schedule
    * @param dbName      the name of the AWS Glue database where the crawler will
store the metadata
    * @param crawlerName the name of the crawler to be created
    */
   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) {
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateCrawler를 참조하세요.

`CreateJob`

다음 코드 예시는 `CreateJob`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Creates a new AWS Glue job.
     *
     * @param glueClient     the AWS Glue client to use for the operation
     * @param jobName        the name of the job to create
     * @param iam            the IAM role to associate with the job
     * @param scriptLocation the location of the script to be used by the job
     * @throws GlueException if there is an error creating the job
     */
    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) {
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateJob을 참조하세요.

`DeleteCrawler`

다음 코드 예시는 `DeleteCrawler`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
```

```
    * Deletes a specific AWS Glue crawler.
    *
    * @param glueClient  the AWS Glue client object
    * @param crawlerName the name of the crawler to be deleted
    * @throws GlueException if an error occurs during the deletion process
    */
   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) {
           throw e;
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteCrawler를 참조하세요.

`DeleteDatabase`

다음 코드 예시는 `DeleteDatabase`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Deletes a AWS Glue Database.
    *
    * @param glueClient   An instance of the AWS Glue client used to interact with
the AWS Glue service.
```

```
     * @param databaseName The name of the database to be deleted.
     * @throws GlueException If an error occurs while deleting the database.
     */
    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) {
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteDatabase를 참조하세요.

`DeleteJob`

다음 코드 예시는 `DeleteJob`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Deletes a Glue job.
     *
     * @param glueClient the Glue client to use for the operation
     * @param jobName    the name of the job to be deleted
     * @throws GlueException if there is an error deleting the job
     */
    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) {
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteJob을 참조하세요.

`GetCrawler`

다음 코드 예시는 `GetCrawler`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Retrieves a specific crawler from the AWS Glue service and waits for it to be
in the "READY" state.
     *
     * @param glueClient  the AWS Glue client used to interact with the Glue service
     * @param crawlerName the name of the crawler to be retrieved
     */
    public static void getSpecificCrawler(GlueClient glueClient, String crawlerName)
throws InterruptedException {
        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) {
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetCrawler를 참조하세요.

`GetDatabase`

다음 코드 예시는 `GetDatabase`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Retrieves the specific database from the AWS Glue service.
     *
     * @param glueClient   an instance of the AWS Glue client used to interact with
the service
     * @param databaseName the name of the database to retrieve
     * @throws GlueException if there is an error retrieving the database from the
AWS Glue service
     */
    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) {
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetDatabase를 참조하세요.

`GetJobRuns`

다음 코드 예시는 `GetJobRuns`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Retrieves the job runs for a given Glue job and prints the status of the job
runs.
     *
     * @param glueClient the Glue client used to make API calls
```

```
     * @param jobName    the name of the Glue job to retrieve the job runs for
     */
    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 e) {
            throw e;
```

```
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetJobRuns를 참조하세요.

`GetTables`

다음 코드 예시는 `GetTables`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Retrieves the names of the tables in the specified Glue database.
     *
     * @param glueClient the Glue client to use for the operation
     * @param dbName     the name of the Glue database to retrieve the table names
from
     * @return the name of the first table retrieved, or an empty string if no
tables were found
     */
    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) {
            throw e;
        }
        return myTableName;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetTables를 참조하세요.

`StartCrawler`

다음 코드 예시는 `StartCrawler`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Starts a specific AWS Glue crawler.
     *
     * @param glueClient  the AWS Glue client to use for the crawler operation
     * @param crawlerName the name of the crawler to start
     * @throws GlueException if there is an error starting the crawler
     */
    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) {
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartCrawler를 참조하세요.

`StartJobRun`

다음 코드 예시는 `StartJobRun`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Starts a job run in AWS Glue.
     *
     * @param glueClient    the AWS Glue client to use for the job run
     * @param jobName       the name of the Glue job to run
     * @param inputDatabase the name of the input database
     * @param inputTable    the name of the input table
     * @param outBucket     the URL of the output S3 bucket
     * @throws GlueException if there is an error starting the job run
     */
    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) {
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartJobRun을 참조하세요.

Java 2.x용 SDK를 사용하는 HealthImaging 예제

다음 코드 예제에서는 HealthImaging과 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업
- 시나리오

작업

CopyImageSet

다음 코드 예시는 CopyImageSet의 사용 방법을 보여줍니다.

SDK for Java 2.x

```
   /**
    * Copy an AWS HealthImaging image set.
    *
    * @param medicalImagingClient  - The AWS HealthImaging client object.
    * @param datastoreId           - The datastore ID.
    * @param imageSetId            - The image set ID.
    * @param latestVersionId       - The version ID.
    * @param destinationImageSetId - The optional destination image set ID, ignored
if null.
    * @param destinationVersionId  - The optional destination version ID, ignored
if null.
    * @param force                 - The force flag.
    * @param subsets               - The optional subsets to copy, ignored if null.
    * @return                      - The image set ID of the copy.
    * @throws MedicalImagingException - Base exception for all service exceptions
thrown by AWS HealthImaging.
    */
   public static String copyMedicalImageSet(MedicalImagingClient
medicalImagingClient,
                                            String datastoreId,
                                            String imageSetId,
                                            String latestVersionId,
                                            String destinationImageSetId,
                                            String destinationVersionId,
                                            boolean force,
                                            Vector<String> subsets) {

       try {
           CopySourceImageSetInformation.Builder copySourceImageSetInformation =
CopySourceImageSetInformation.builder()
                   .latestVersionId(latestVersionId);

           // Optionally copy a subset of image instances.
           if (subsets != null) {
```

```
                String subsetInstanceToCopy = getCopiableAttributesJSON(imageSetId,
subsets);
                copySourceImageSetInformation.dicomCopies(MetadataCopies.builder()
                        .copiableAttributes(subsetInstanceToCopy)
                        .build());
            }

            CopyImageSetInformation.Builder copyImageSetBuilder =
CopyImageSetInformation.builder()
                    .sourceImageSet(copySourceImageSetInformation.build());

            // Optionally designate a destination image set.
            if (destinationImageSetId != null) {
                copyImageSetBuilder =
copyImageSetBuilder.destinationImageSet(CopyDestinationImageSet.builder()
                        .imageSetId(destinationImageSetId)
                        .latestVersionId(destinationVersionId)
                        .build());
            }

            CopyImageSetRequest copyImageSetRequest = CopyImageSetRequest.builder()
                    .datastoreId(datastoreId)
                    .sourceImageSetId(imageSetId)
                    .copyImageSetInformation(copyImageSetBuilder.build())
                    .force(force)
                    .build();

            CopyImageSetResponse response =
medicalImagingClient.copyImageSet(copyImageSetRequest);

            return response.destinationImageSetProperties().imageSetId();
        } catch (MedicalImagingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            throw e;
        }
    }
```

복사 가능한 속성을 만드는 유틸리티 함수입니다.

```
    /**
     * Create a JSON string of copiable image instances.
```

```
     *
     * @param imageSetId - The image set ID.
     * @param subsets    - The subsets to copy.
     * @return A JSON string of copiable image instances.
     */
    private static String getCopiableAttributesJSON(String imageSetId,
Vector<String> subsets) {
        StringBuilder subsetInstanceToCopy = new StringBuilder(
                """
                        {
                          "SchemaVersion": 1.1,
                          "Study": {
                            "Series": {
                                "
                                 """
        );

        subsetInstanceToCopy.append(imageSetId);

        subsetInstanceToCopy.append(
                """
                                ": {
                                "Instances": {
                        """
        );

        for (String subset : subsets) {
            subsetInstanceToCopy.append('"' + subset + "\": {},");
        }
        subsetInstanceToCopy.deleteCharAt(subsetInstanceToCopy.length() - 1);
        subsetInstanceToCopy.append("""
                         }
                       }
                     }
                   }
                 }
                 """);
        return subsetInstanceToCopy.toString();
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CopyImageSet를 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

`CreateDatastore`

다음 코드 예시는 `CreateDatastore`의 사용 방법을 보여줍니다.

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 "";
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateDatastore를 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

`DeleteDatastore`

다음 코드 예시는 `DeleteDatastore`의 사용 방법을 보여줍니다.

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);
       }
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteDatastore를 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

`DeleteImageSet`

다음 코드 예시는 `DeleteImageSet`의 사용 방법을 보여줍니다.

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);
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteImageSet를 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

`GetDICOMImportJob`

다음 코드 예시는 `GetDICOMImportJob`의 사용 방법을 보여줍니다.

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;
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 GetDICOMImportJob을 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

`GetDatastore`

다음 코드 예시는 `GetDatastore`의 사용 방법을 보여줍니다.

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;
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 GetDatastore를 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

`GetImageFrame`

다음 코드 예시는 `GetImageFrame`의 사용 방법을 보여줍니다.

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);
                }
        }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 GetImageFrame을 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

`GetImageSet`

다음 코드 예시는 `GetImageSet`의 사용 방법을 보여줍니다.

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;
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 GetImageSet를 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

`GetImageSetMetadata`

다음 코드 예시는 `GetImageSetMetadata`의 사용 방법을 보여줍니다.

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);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetImageSetMetadata를 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

`ListDICOMImportJobs`

다음 코드 예시는 `ListDICOMImportJobs`의 사용 방법을 보여줍니다.

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<>();
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListDICOMImportJobs를 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

`ListDatastores`

다음 코드 예시는 `ListDatastores`의 사용 방법을 보여줍니다.

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;
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListDatastores를 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

`ListImageSetVersions`

다음 코드 예시는 `ListImageSetVersions`의 사용 방법을 보여줍니다.

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;
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListImageSetVersions를 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

`ListTagsForResource`

다음 코드 예시는 `ListTagsForResource`의 사용 방법을 보여줍니다.

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;
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListTagsForResource를 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

`SearchImageSets`

다음 코드 예시는 `SearchImageSets`의 사용 방법을 보여줍니다.

SDK for Java 2.x

이미지 세트 검색을 위한 유틸리티 함수.

```
   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;
    }
```

사용 사례 #1: EQUAL 연산자.

```
        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();
        }
```

사용 사례 #2: DICOMStudyDate 및 DICOMStudyTime을 사용한 BETWEEN 연산자.

```
        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();
        }
```

사용 사례 #3: createdAt을 사용한 BETWEEN 연산자. 시간 연구가 이전에 지속되었습니다.

```
        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();
        }
```

사용 사례 4: DICOMSeriesInstanceUID 필드에는 EQUAL 연산자를 적용하고 updatedAt 필드에는 BETWEEN 연산자를 적용하며, 응답은 updatedAt 필드를 기준으로 오름차순(ASC) 정렬합니다.

```
        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();
        }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 SearchImageSets를 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

`StartDICOMImportJob`

다음 코드 예시는 `StartDICOMImportJob`의 사용 방법을 보여줍니다.

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 "";
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 StartDICOMImportJob을 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

`TagResource`

다음 코드 예시는 `TagResource`의 사용 방법을 보여줍니다.

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);
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 TagResource를 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

`UntagResource`

다음 코드 예시는 `UntagResource`의 사용 방법을 보여줍니다.

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);
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 UntagResource를 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

`UpdateImageSetMetadata`

다음 코드 예시는 `UpdateImageSetMetadata`의 사용 방법을 보여줍니다.

SDK for Java 2.x

```
    /**
     * Update the metadata of an AWS HealthImaging image set.
     *
     * @param medicalImagingClient - The AWS HealthImaging client object.
     * @param datastoreId          - The datastore ID.
     * @param imageSetId           - The image set ID.
     * @param versionId            - The version ID.
     * @param metadataUpdates      - A MetadataUpdates object containing the
updates.
```

```
    * @param force                - The force flag.
    * @throws MedicalImagingException - Base exception for all service exceptions
thrown by AWS HealthImaging.
    */
   public static void updateMedicalImageSetMetadata(MedicalImagingClient
medicalImagingClient,
                                                    String datastoreId,
                                                    String imageSetId,
                                                    String versionId,
                                                    MetadataUpdates
metadataUpdates,
                                                    boolean force) {
       try {
           UpdateImageSetMetadataRequest updateImageSetMetadataRequest =
UpdateImageSetMetadataRequest
                   .builder()
                   .datastoreId(datastoreId)
                   .imageSetId(imageSetId)
                   .latestVersionId(versionId)
                   .updateImageSetMetadataUpdates(metadataUpdates)
                   .force(force)
                   .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());
           throw e;
       }
   }
```

사용 사례 1: 속성을 삽입하거나 업데이트합니다.

```
               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, force);
```

사용 사례 2: 속성을 제거합니다.

```
            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, force);
```

사용 사례 3: 인스턴스를 제거합니다.

```
                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, force);
```

사용 사례 4: 이전 버전으로 되돌립니다.

```
                // In this case, revert to previous version.
                String revertVersionId =
Integer.toString(Integer.parseInt(versionid) - 1);
                MetadataUpdates metadataRemoveUpdates = MetadataUpdates.builder()
                        .revertToVersionId(revertVersionId)
                        .build();
                updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
imagesetId,
                        versionid, metadataRemoveUpdates, force);
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateImageSetMetadata를 참조하세요.

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

시나리오

데이터 저장소에 태그 지정

다음 코드 예제에서는 HealthImaging 데이터 스토어에 태그를 지정하는 방법을 보여줍니다.

SDK for Java 2.x

데이터 스토어에 태깅하려면.

```
                final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

                TagResource.tagMedicalImagingResource(medicalImagingClient,
 datastoreArn,
                                ImmutableMap.of("Deployment", "Development"));
```

리소스에 태그를 지정하는 유틸리티 함수.

```
   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);
        }
    }
```

데이터 스토어의 태그를 나열하려면.

```
                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());
                }
```

리소스의 태그를 나열하는 유틸리티 함수입니다.

```
    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;
    }
```

데이터 스토어에 태그 지정을 해제하려면.

```
                final String datastoreArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012";

                UntagResource.untagMedicalImagingResource(medicalImagingClient,
 datastoreArn,
                                Collections.singletonList("Deployment"));
```

리소스의 태그를 해제하는 유틸리티 함수.

```
   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);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하십시오.
  - ListTagsForResource
  - TagResource
  - UntagResource

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

이미지 세트 태그 지정

다음 코드 예제는 HealthImaging 이미지 세트를 태그하는 방법을 보여줍니다.

SDK for Java 2.x

이미지 세트에 태그를 지정하려면.

```
                final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";

                TagResource.tagMedicalImagingResource(medicalImagingClient,
 imageSetArn,
                                ImmutableMap.of("Deployment", "Development"));
```

리소스에 태그를 지정하는 유틸리티 함수.

```
   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);
        }
```

```
    }
```

이미지 세트의 태그를 나열하려면.

```
               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());
               }
```

리소스의 태그를 나열하는 유틸리티 함수입니다.

```
   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;
   }
```

이미지 세트의 태그를 해제하려면.

```
                final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";

                UntagResource.untagMedicalImagingResource(medicalImagingClient,
 imageSetArn,
                                Collections.singletonList("Deployment"));
```

리소스의 태그를 해제하는 유틸리티 함수.

```
   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);
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하십시오.
  - ListTagsForResource
  - TagResource
  - UntagResource

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Java 2.x용 SDK를 사용하는 IAM 예제

다음 코드 예제에서는 IAM과 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시작하기

Hello IAM

다음 코드 예제에서는 IAM 사용을 시작하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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());
        });
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListPolicies를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 사용자를 생성하고 역할을 수임하는 방법을 보여줍니다.

> **Warning**
>
> 보안 위험을 방지하려면 목적별 소프트웨어를 개발하거나 실제 데이터로 작업할 때 IAM 사용자를 인증에 사용하지 마세요. 대신 AWS IAM Identity Center과 같은 보안 인증 공급자를 통한 페더레이션을 사용하십시오.

- 권한이 없는 사용자를 생성합니다.
- 계정에 대한 Amazon S3 버킷을 나열할 수 있는 권한을 부여하는 역할을 생성합니다.
- 사용자가 역할을 수임할 수 있도록 정책을 추가합니다.
- 역할을 수임하고 임시 자격 증명 정보를 사용하여 S3 버킷을 나열한 후 리소스를 정리합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

IAM 사용자 작업을 래핑하는 함수를 생성합니다.

```
/*
  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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하십시오.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy

- PutUserPolicy

작업

`AttachRolePolicy`

다음 코드 예시는 `AttachRolePolicy`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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");
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 AttachRolePolicy를 참조하세요.

`CreateAccessKey`

다음 코드 예시는 `CreateAccessKey`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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 "";
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateAccessKey를 참조하세요.

`CreateAccountAlias`

다음 코드 예시는 `CreateAccountAlias`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateAccountAlias를 참조하세요.

`CreatePolicy`

다음 코드 예시는 `CreatePolicy`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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 "";
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreatePolicy를 참조하세요.

`CreateRole`

다음 코드 예시는 `CreateRole`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateRole을 참조하세요.

`CreateUser`

다음 코드 예시는 `CreateUser`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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 "";
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateUser를 참조하세요.

`DeleteAccessKey`

다음 코드 예시는 `DeleteAccessKey`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteAccessKey를 참조하세요.

`DeleteAccountAlias`

다음 코드 예시는 `DeleteAccountAlias`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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");
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteAccountAlias를 참조하세요.

`DeletePolicy`

다음 코드 예시는 `DeletePolicy`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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");
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeletePolicy를 참조하세요.

`DeleteUser`

다음 코드 예시는 `DeleteUser`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteUser를 참조하세요.

`DetachRolePolicy`

다음 코드 예시는 `DetachRolePolicy`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DetachRolePolicy를 참조하세요.

`ListAccessKeys`

다음 코드 예시는 `ListAccessKeys`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
    }
  }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListAccessKeys를 참조하세요.

`ListAccountAliases`

다음 코드 예시는 `ListAccountAliases`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListAccountAliases를 참조하세요.

`ListUsers`

다음 코드 예시는 `ListUsers`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
```

```
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListUsers를 참조하세요.

`UpdateAccessKey`

다음 코드 예시는 `UpdateAccessKey`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateAccessKey를 참조하세요.

`UpdateUser`

다음 코드 예시는 `UpdateUser`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateUser를 참조하세요.

시나리오

복원력이 뛰어난 서비스 구축 및 관리

다음 코드 예제에서는 책, 영화, 노래 추천을 반환하는 로드 밸런싱 웹 서비스를 만드는 방법을 보여줍니다. 이 예제에서는 서비스가 장애에 대응하는 방법과 장애 발생 시 복원력을 높이기 위해 서비스를 재구성하는 방법을 보여줍니다.

- Amazon EC2 Auto Scaling 그룹을 사용하여 시작 템플릿을 기반으로 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스를 생성하고 인스턴스 수를 지정된 범위 내로 유지합니다.
- Elastic Load Balancing으로 HTTP 요청을 처리하고 배포합니다.
- Auto Scaling 그룹의 인스턴스 상태를 모니터링하고 요청을 정상 인스턴스로만 전달합니다.
- 각 EC2 인스턴스에서 Python 웹 서버를 실행하여 HTTP 요청을 처리합니다. 웹 서버는 추천 및 상태 확인으로 응답합니다.
- Amazon DynamoDB 테이블을 사용하여 추천 서비스를 시뮬레이션합니다.
- AWS Systems Manager 파라미터를 업데이트하여 요청 및 상태 확인에 대한 웹 서버 응답을 제어합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
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);
    }
}
```

Auto Scaling과 Amazon EC2 작업을 래핑하는 클래스를 생성합니다.

```
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.");
    }
}
```

ELB 작업을 래핑하는 클래스를 생성합니다.

```
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)
                    .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 "";
    }
}
```

DynamoDB를 사용하여 추천 서비스를 시뮬레이션하는 클래스를 생성합니다.

```
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);
    }
}
```

Systems Manager 작업을 래핑하는 클래스를 생성합니다.

```
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);
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - 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

IAM 정책 빌더 API 작업

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 객체 지향 API를 사용하여 IAM 정책을 생성합니다.
- IAM 서비스에 IAM 정책 빌더 API를 사용합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

예제에서는 다음 가져오기를 사용합니다.

```
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;
```

시간 기반 정책을 생성합니다.

```
        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());
        }
```

여러 조건이 포함된 정책을 생성합니다.

```
        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());
        }
```

정책에 위탁자를 사용합니다.

```
        public String specifyPrincipalsExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.DENY)
                                                .addAction("s3:*")
                                                .addPrincipal(IamPrincipal.ALL)
                                                .addResource("arn:aws:s3:::amzn-s3-
demo-bucket/*")
                                                .addResource("arn:aws:s3:::amzn-s3-
demo-bucket")
                                                .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());
        }
```

교차 계정 액세스를 허용합니다.

```
        public String allowCrossAccountAccessExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.ALLOW)
                                                .addPrincipal(IamPrincipalType.AWS,
 "111122223333")
                                                .addAction("s3:PutObject")
                                                .addResource("arn:aws:s3:::amzn-s3-
demo-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());
        }
```

IamPolicy를 빌드하고 업로드합니다.

```
        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());
        }
```

IamPolicy를 다운로드하고 사용합니다.

```
        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());
        }
```

- 자세한 정보는 AWS SDK for Java 2.x 개발자 안내서를 참조하세요.
- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - CreatePolicy
  - GetPolicy
  - GetPolicyVersion

SDK for Java 2.x를 사용한 Amazon Inspector 예제

다음 코드 예제에서는 Amazon Inspector에서를 사용하여 작업을 수행하고 일반적인 시나리오 AWS SDK for Java 2.x 를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시작하기
- 기본 사항
- 작업

시작하기

Hello

다음 코드 예제에서는 를 사용하여 시작하는 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloInspector {
    private static final Logger logger =
 LoggerFactory.getLogger(HelloInspector.class);

    public static void main(String[] args) {
        logger.info("Hello Amazon Inspector!");

        try (Inspector2Client inspectorClient = Inspector2Client.builder().build())
 {

            logger.info("Listing member accounts for this Inspector administrator
 account...");
            listMembers(inspectorClient);

            logger.info("The Hello Inspector example completed successfully.");

        } catch (Inspector2Exception e) {
            logger.error("Error: {}", e.getMessage());
            logger.info("Troubleshooting:");
            logger.info("1. Verify AWS credentials are configured");
```

```
            logger.info("2. Check IAM permissions for Inspector2");
            logger.info("3. Ensure Inspector2 is enabled in your account");
            logger.info("4. Verify you're using a supported region");
        }
    }

    /**
     * Lists all member accounts associated with the current Inspector administrator
 account.
     *
     * @param inspectorClient The Inspector2Client used to interact with AWS
 Inspector.
     */
    public static void listMembers(Inspector2Client inspectorClient) {
        try {
            ListMembersRequest request = ListMembersRequest.builder()
                    .maxResults(50) // optional: limit results
                    .build();

            ListMembersResponse response = inspectorClient.listMembers(request);
            List<Member> members = response.members();

            if (members == null || members.isEmpty()) {
                logger.info("No member accounts found for this Inspector
 administrator account.");
                return;
            }

            logger.info("Found {} member account(s):", members.size());
            for (Member member : members) {
                logger.info(" - Account ID: {}, Status: {}",
                        member.accountId(),
                        member.relationshipStatusAsString());
            }

        } catch (Inspector2Exception e) {
            logger.error("Failed to list members: {}",
 e.awsErrorDetails().errorMessage());
        }
    }
}
```

- API 세부 정보는 API 참조의 ListMembersAWS SDK for Java 2.x 를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- Inspector 계정 상태를 확인합니다.
- Inspector가 활성화되어 있는지 확인합니다.
- 보안 조사 결과를 분석합니다.
- 스캔 적용 범위를 확인합니다.
- 결과 필터를 생성합니다.
- 기존 필터를 나열합니다.
- 사용량 및 비용을 확인합니다.
- 적용 범위 통계를 가져옵니다.
- 필터를 삭제합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

기능을 보여주는 대화형 시나리오를 실행합니다.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class InspectorScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");
```

```
   private static final Logger logger =
LoggerFactory.getLogger(InspectorScenario.class);
   private static final Scanner scanner = new Scanner(System.in);

   public static void main(String[] args) {
       InspectorActions inspectorActions = new InspectorActions();
       logger.info("Amazon Inspector Basics Scenario");

       logger.info("""
                   Amazon Inspector is a security assessment service provided by
Amazon Web Services (AWS) that helps
                   improve the security and compliance of applications deployed on
AWS. It automatically assesses
                   applications for vulnerabilities or deviations from best
practices. By leveraging Amazon Inspector,
                   users can gain insights into the overall security state of their
application and identify potential
                   security risks.

                   This service operates by conducting both network and host-based
assessments, allowing it to detect a
                   wide range of security issues, including those related to
operating systems, network configurations,
                   and application dependencies.
                   """);

       waitForInputToContinue();

       try {
           runScenario(inspectorActions);

           logger.info("");
           logger.info("Scenario completed successfully!");
           logger.info("");
           logger.info("What you learned:");
           logger.info("  - How to check Inspector account status");
           logger.info("  - How to enable Inspector");
           logger.info("  - How to list and analyze findings");
           logger.info("  - How to check coverage information");
           logger.info("  - How to create and manage filters");
           logger.info("  - How to track usage and costs");
           logger.info("  - How to clean up resources");
           logger.info("");
```

```
        } catch (Exception ex) {
            logger.error("Scenario failed due to unexpected error: {}",
ex.getMessage(), ex);

        } finally {
            scanner.close();
            logger.info("Exiting...");
        }
    }

    /**
     * Runs the Inspector scenario in a step-by-step sequence.
     *
     * All InspectorActions methods are asynchronous and return CompletableFutures.
     * Each step ends with .join(). Any async exception thrown during .join() will
bubble up
     *
     */
    public static void runScenario(InspectorActions actions) {
        String filterArn = null;
        boolean inspectorEnabled = false;

        try {
            // Step 1
            logger.info(DASHES);
            logger.info("Step 1: Checking Inspector account status...");
            String status = actions.getAccountStatusAsync().join();
            logger.info(status);
            waitForInputToContinue();

            // Step 2
            logger.info(DASHES);
            logger.info("Step 2: Enabling Inspector...");
            String message = actions.enableInspectorAsync(null).join();
            logger.info(message);
            inspectorEnabled = true;  // track that Inspector was enabled
            waitForInputToContinue();

            // Step 3
            logger.info(DASHES);
            logger.info("Step 3: Listing LOW severity findings...");

            // Call the service method
```

```
            List<String> allFindings =
actions.listLowSeverityFindingsAsync().join();

            if (!allFindings.isEmpty()) {
                // Only proceed if there are findings
                String lastArn = allFindings.get(allFindings.size() - 1);
                logger.info("Look up details on: {}", lastArn);
                waitForInputToContinue();
                String details = actions.getFindingDetailsAsync(lastArn).join();
                logger.info(details);
            } else {
                logger.info("No LOW severity findings found.");
            }

            waitForInputToContinue();

            // Step 4
            logger.info(DASHES);
            logger.info("Step 4: Listing coverage...");
            String coverage = actions.listCoverageAsync(5).join();
            logger.info(coverage);
            waitForInputToContinue();

            // Step 5
            logger.info(DASHES);
            logger.info("Step 5: Creating filter...");
            String filterName = "suppress-low-" + System.currentTimeMillis();
            filterArn = actions.createLowSeverityFilterAsync(filterName, "Suppress
low severity findings").join();
            logger.info("Created filter: {}", filterArn);
            waitForInputToContinue();

            // Step 6
            logger.info(DASHES);
            logger.info("Step 6: Listing filters...");
            String filters = actions.listFiltersAsync(10).join();
            logger.info(filters);
            waitForInputToContinue();

            // Step 7
            logger.info(DASHES);
            logger.info("Step 7: Usage totals...");
            String usage = actions.listUsageTotalsAsync(null, 10).join();
            logger.info(usage);
```

```
            waitForInputToContinue();

            // Step 8
            logger.info(DASHES);
            logger.info("Step 8: Coverage statistics...");
            String stats = actions.listCoverageStatisticsAsync().join();
            logger.info(stats);
            waitForInputToContinue();

            // Step 9
            logger.info(DASHES);
            logger.info("Step 9: Delete filter?");
            logger.info("Filter ARN: {}", filterArn);
            logger.info("Delete the filter and disable Inspector? (y/n)");

            if (scanner.nextLine().trim().equalsIgnoreCase("y")) {
                actions.deleteFilterAsync(filterArn).join();
                logger.info("Filter deleted.");
                String disableMsg = actions.disableInspectorAsync(null).join();
                logger.info(disableMsg);
                inspectorEnabled = false; // track that Inspector was disabled
            }

            waitForInputToContinue();

        } catch (Exception ex) {
            logger.error("Scenario encountered an error: {}", ex.getMessage(), ex);
            // Rethrow the exception
            throw ex;

        } finally {
            // Cleanup in case of an exception
            if (filterArn != null) {
                try {
                    actions.deleteFilterAsync(filterArn).join();
                    logger.info("Cleanup: Filter deleted.");
                } catch (Exception e) {
                    logger.warn("Failed to delete filter during cleanup: {}",
e.getMessage(), e);
                }
            }

            if (inspectorEnabled) {
                try {
```

```
                    actions.disableInspectorAsync(null).join();
                    logger.info("Cleanup: Inspector disabled.");
                } catch (Exception e) {
                    logger.warn("Failed to disable Inspector during cleanup: {}",
 e.getMessage(), e);
                }
            }
        }
    }

    // Utility Method
    private static void waitForInputToContinue() {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' to continue:");
            String input = scanner.nextLine().trim();
            if (input.equalsIgnoreCase("c")) break;
            logger.info("Invalid input, try again.");
        }
    }
}
```

SDK 메서드의 래퍼 클래스입니다.

```
public class InspectorActions {
    private static Inspector2AsyncClient inspectorAsyncClient;
    private static final Logger logger =
 LoggerFactory.getLogger(InspectorActions.class);

    private static Inspector2AsyncClient getAsyncClient() {
        if (inspectorAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                    .maxConcurrency(100)
                    .connectionTimeout(Duration.ofSeconds(60))
                    .readTimeout(Duration.ofSeconds(60))
                    .writeTimeout(Duration.ofSeconds(60))
                    .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
```

```
                    .retryStrategy(RetryMode.STANDARD)
                    .build();

            inspectorAsyncClient = Inspector2AsyncClient.builder()
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
        return inspectorAsyncClient;
    }


    /**
     * Enables AWS Inspector for the provided account(s) and default resource types.
     *
     * @param accountIds Optional list of AWS account IDs.
     */
    public CompletableFuture<String> enableInspectorAsync(List<String> accountIds) {

        // The resource types to enable.
        List<ResourceScanType> resourceTypes = List.of(
                ResourceScanType.EC2,
                ResourceScanType.ECR,
                ResourceScanType.LAMBDA,
                ResourceScanType.LAMBDA_CODE
        );

        // Build the request.
        EnableRequest.Builder requestBuilder = EnableRequest.builder()
                .resourceTypes(resourceTypes);

        if (accountIds != null && !accountIds.isEmpty()) {
            requestBuilder.accountIds(accountIds);
        }

        EnableRequest request = requestBuilder.build();
        return getAsyncClient().enable(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ValidationException) {
                            throw new CompletionException(
                                    "Inspector may already be enabled for this
account: %s".formatted(cause.getMessage()),
```

```
                                    cause
                            );

                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;
                            throw new CompletionException(
                                    "AWS Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to enable Inspector:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    StringBuilder summary = new StringBuilder("Enable results:\n");

                    if (response.accounts() == null ||
response.accounts().isEmpty()) {
                        summary.append("Inspector may already be enabled for all
target accounts.");
                        return summary.toString();
                    }

                    for (Account account : response.accounts()) {
                        String accountId = account.accountId() != null ?
account.accountId() : "Unknown";
                        String status = account.status() != null ?
account.statusAsString() : "Unknown";
                        summary.append(" • Account: ").append(accountId)
                                .append(" # Status: ").append(status).append("\n");
                    }

                    return summary.toString();
                });
    }
```

```
   /**
    * Retrieves and prints the coverage statistics using a paginator.
    */
   public CompletableFuture<String> listCoverageStatisticsAsync() {
       ListCoverageStatisticsRequest request =
ListCoverageStatisticsRequest.builder()
               .build();

       return getAsyncClient().listCoverageStatistics(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();

                       if (cause instanceof ValidationException) {
                           throw new CompletionException(
                                   "Validation error listing coverage statistics:
%s".formatted(cause.getMessage()),
                                   cause
                           );
                       }

                       if (cause instanceof Inspector2Exception) {
                           Inspector2Exception e = (Inspector2Exception) cause;

                           throw new CompletionException(
                                   "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                   e
                           );
                       }

                       throw new CompletionException(
                               "Unexpected error listing coverage statistics:
%s".formatted(exception.getMessage()),
                               exception
                       );
                   }
               })
               .thenApply(response -> {
                   List<Counts> countsList = response.countsByGroup();
                   StringBuilder sb = new StringBuilder();

                   if (countsList == null || countsList.isEmpty()) {
```

```
                        sb.append("No coverage statistics available.\n");
                        return sb.toString();
                    }

                    sb.append("Coverage Statistics:\n");

                    for (Counts c : countsList) {
                        sb.append("  Group: ").append(c.groupKey()).append("\n")
                                .append("    Total Count:
").append(c.count()).append("\n\n");
                    }

                    return sb.toString();
                });
    }


    /**
     * Asynchronously lists Inspector2 usage totals using a paginator.
     *
     * @param accountIds optional list of account IDs
     * @param maxResults maximum results per page
     * @return CompletableFuture completed with formatted summary text
     */
    public CompletableFuture<String> listUsageTotalsAsync(
            List<String> accountIds,
            int maxResults) {

        logger.info("Starting usage totals paginator…");

        ListUsageTotalsRequest.Builder builder = ListUsageTotalsRequest.builder()
                .maxResults(maxResults);

        if (accountIds != null && !accountIds.isEmpty()) {
            builder.accountIds(accountIds);
        }

        ListUsageTotalsRequest request = builder.build();
        ListUsageTotalsPublisher paginator =
getAsyncClient().listUsageTotalsPaginator(request);
        StringBuilder summaryBuilder = new StringBuilder();

        return paginator.subscribe(response -> {
                    if (response.totals() != null && !response.totals().isEmpty()) {
```

```
                        response.totals().forEach(total -> {
                            if (total.usage() != null) {
                                total.usage().forEach(usage -> {
                                    logger.info("Usage: {} = {}",
usage.typeAsString(), usage.total());
                                    summaryBuilder.append(usage.typeAsString())
                                            .append(": ")
                                            .append(usage.total())
                                            .append("\n");
                                });
                            }
                        });
                    } else {
                        logger.info("Page contained no usage totals.");
                    }
                }).thenRun(() -> logger.info("Successfully listed usage totals."))
                .thenApply(v -> {
                    String summary = summaryBuilder.toString();
                    return summary.isEmpty() ? "No usage totals found." : summary;
                }).exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ValidationException ve) {
                        throw new CompletionException(
                                "Validation error listing usage totals:
%s".formatted(ve.getMessage()),
                                ve
                        );
                    }

                    throw new CompletionException("Failed to list usage totals",
cause);
                });
    }



    /**
     * Retrieves the account status using the Inspector2Client.
     */
    public CompletableFuture<String> getAccountStatusAsync() {
        BatchGetAccountStatusRequest request =
BatchGetAccountStatusRequest.builder()
                .accountIds(Collections.emptyList())
```

```
                .build();

        return getAsyncClient().batchGetAccountStatus(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof AccessDeniedException) {
                            throw new CompletionException(
                                    "You do not have sufficient access:
%s".formatted(cause.getMessage()),
                                    cause
                            );

                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;

                            throw new CompletionException(
                                    "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    e
                            );
                        }

                        throw new CompletionException(
                                "Unexpected error getting account status:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {

                    StringBuilder sb = new StringBuilder();
                    List<AccountState> accounts = response.accounts();

                    if (accounts == null || accounts.isEmpty()) {
                        sb.append("No account status returned.\n");
                        return sb.toString();
                    }

                    sb.append("Inspector Account Status:\n");
                    for (AccountState account : accounts) {
```

```
                        String accountId = account.accountId() != null
                                ? account.accountId()
                                : "Unknown";

                        sb.append("  Account ID: ").append(accountId).append("\n");

                        // Overall account state
                        if (account.state() != null && account.state().status() !=
null) {
                            sb.append("  Overall State: ")
                                    .append(account.state().status())
                                    .append("\n");
                        } else {
                            sb.append("  Overall State: Unknown\n");
                        }

                        // Resource state (only status available)
                        ResourceState resources = account.resourceState();
                        if (resources != null) {
                            sb.append("  Resource Status: available\n");
                        }

                        sb.append("\n");
                    }

                    return sb.toString();
                });
    }


    /**
     * Asynchronously lists Inspector2 filters using a paginator.
     *
     * @param maxResults maximum filters per page (nullable)
     * @return CompletableFuture completed with summary text
     */
    public CompletableFuture<String> listFiltersAsync(Integer maxResults) {
        logger.info("Starting async filters paginator…");

        ListFiltersRequest.Builder builder = ListFiltersRequest.builder();
        if (maxResults != null) {
            builder.maxResults(maxResults);
        }
```

```
        ListFiltersRequest request = builder.build();

        // Paginator from SDK
        ListFiltersPublisher paginator =
getAsyncClient().listFiltersPaginator(request);
        StringBuilder collectedFilterIds = new StringBuilder();

        return paginator.subscribe(response -> {
            response.filters().forEach(filter -> {
                logger.info("Filter: " + filter.arn());
                collectedFilterIds.append(filter.arn()).append("\n");
            });
        }).thenApply(v -> {
            String result = collectedFilterIds.toString();
            logger.info("Successfully listed all filters.");
            return result.isEmpty() ? "No filters found." : result;
        }).exceptionally(ex -> {
            Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

            if (cause instanceof ValidationException ve) {
                throw new CompletionException(
                        "Validation error listing filters:
%s".formatted(ve.getMessage()),
                        ve
                );
            }

            throw new RuntimeException("Failed to list filters", ex);
        });
    }


    /**
     * Creates a new LOW severity filter in AWS Inspector2 to suppress findings.
     *
     * @param filterName  the name of the filter to create
     * @param description a descriptive string explaining the purpose of the filter
     * @return a CompletableFuture that completes with the ARN of the created filter
     * @throws CompletionException wraps any validation, Inspector2 service, or
unexpected errors
     */
    public CompletableFuture<String> createLowSeverityFilterAsync(
            String filterName,
```

```
            String description) {

        // Define a filter to match LOW severity findings.
        StringFilter severityFilter = StringFilter.builder()
                .value(Severity.LOW.toString())
                .comparison(StringComparison.EQUALS)
                .build();

        // Create filter criteria.
        FilterCriteria filterCriteria = FilterCriteria.builder()
                .severity(Collections.singletonList(severityFilter))
                .build();

        // Build the filter creation request.
        CreateFilterRequest request = CreateFilterRequest.builder()
                .name(filterName)
                .filterCriteria(filterCriteria)
                .action(FilterAction.SUPPRESS)
                .description(description)
                .build();

        return getAsyncClient().createFilter(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ValidationException ve) {
                            throw new CompletionException(
                                    "Validation error creating filter:
%s".formatted(ve.getMessage()),
                                    ve
                            );
                        }

                        if (cause instanceof Inspector2Exception e) {
                            throw new CompletionException(
                                    "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    e
                            );
                        }

                        // Unexpected async error
```

```
                        throw new CompletionException(
                                "Unexpected error creating filter:
 %s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                // Extract and return the ARN of the created filter.
                .thenApply(CreateFilterResponse::arn);
    }


    /**
     * Lists all AWS Inspector findings of LOW severity asynchronously.
     *
     * @return CompletableFuture containing a List of finding ARNs.
     * Returns an empty list if no LOW severity findings are found.
     */
    public CompletableFuture<ArrayList<String>> listLowSeverityFindingsAsync() {
        logger.info("Starting async LOW severity findings paginator…");

        // Build a filter criteria for LOW severity.
        StringFilter severityFilter = StringFilter.builder()
                .value(Severity.LOW.toString())
                .comparison(StringComparison.EQUALS)
                .build();

        FilterCriteria filterCriteria = FilterCriteria.builder()
                .severity(Collections.singletonList(severityFilter))
                .build();

        // Build the request.
        ListFindingsRequest request = ListFindingsRequest.builder()
                .filterCriteria(filterCriteria)
                .build();

        ListFindingsPublisher paginator =
 getAsyncClient().listFindingsPaginator(request);
        List<String> allArns = Collections.synchronizedList(new ArrayList<>());

        return paginator.subscribe(response -> {
                    if (response.findings() != null && !
response.findings().isEmpty()) {
                        response.findings().forEach(finding -> {
```

```
                            logger.info("Finding ARN: {}", finding.findingArn());
                            allArns.add(finding.findingArn());
                        });
                    } else {
                        logger.info("Page contained no findings.");
                    }
                })
                .thenRun(() -> logger.info("Successfully listed all LOW severity
findings."))
                .thenApply(v -> new ArrayList<>(allArns)) // Return list instead of
a formatted string
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;
                    if (cause instanceof ValidationException ve) {
                        throw new CompletionException(
                                "Validation error listing LOW severity findings:
%s".formatted(ve.getMessage()),
                                ve
                        );
                    }
                    throw new RuntimeException("Failed to list LOW severity
findings", ex);
                });
    }



    /**
     * Lists AWS Inspector2 coverage details for scanned resources using a
paginator.
     *
     * @param maxResults Maximum number of resources to return.
     */
    public CompletableFuture<String> listCoverageAsync(int maxResults) {
        ListCoverageRequest initialRequest = ListCoverageRequest.builder()
                .maxResults(maxResults)
                .build();

        ListCoveragePublisher paginator =
getAsyncClient().listCoveragePaginator(initialRequest);
        StringBuilder summary = new StringBuilder();

        return paginator.subscribe(response -> {
            List<CoveredResource> coveredResources = response.coveredResources();
```

```
            if (coveredResources == null || coveredResources.isEmpty()) {
                summary.append("No coverage information available for this page.
\n");
                return;
            }

            Map<String, List<CoveredResource>> byType = coveredResources.stream()

 .collect(Collectors.groupingBy(CoveredResource::resourceTypeAsString));

            byType.forEach((type, list) ->
                    summary.append("  ").append(type)
                            .append(": ").append(list.size())
                            .append(" resource(s)\n")
            );

            // Include up to 3 sample resources per page
            for (int i = 0; i < Math.min(coveredResources.size(), 3); i++) {
                CoveredResource r = coveredResources.get(i);
                summary.append("  - ").append(r.resourceTypeAsString())
                        .append(": ").append(r.resourceId()).append("\n");
                summary.append("    Scan Type:
 ").append(r.scanTypeAsString()).append("\n");
                if (r.scanStatus() != null) {
                    summary.append("    Status:
 ").append(r.scanStatus().statusCodeAsString()).append("\n");
                }
                if (r.accountId() != null) {
                    summary.append("    Account ID:
 ").append(r.accountId()).append("\n");
                }
                summary.append("\n");
            }

        }).thenApply(v -> {
            if (summary.length() == 0) {
                return "No coverage information found across all pages.";
            } else {
                return "Coverage Information:\n" + summary.toString();
            }
        }).exceptionally(ex -> {
            Throwable cause = ex.getCause();
            if (cause instanceof ValidationException) {
```

```
                throw new CompletionException(
                        "Validation error listing coverage: " + cause.getMessage(),
cause);
            } else if (cause instanceof Inspector2Exception e) {
                throw new CompletionException(
                        "Inspector2 service error: " +
e.awsErrorDetails().errorMessage(), e);
            }
            throw new CompletionException("Unexpected error listing coverage: " +
ex.getMessage(), ex);
        });
    }


    /**
     * Deletes an AWS Inspector2 filter.
     *
     * @param filterARN The ARN of the filter to delete.
     */
    public CompletableFuture<Void> deleteFilterAsync(String filterARN) {
        return getAsyncClient().deleteFilter(
                        DeleteFilterRequest.builder()
                                .arn(filterARN)
                                .build()
                )
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ResourceNotFoundException rnfe) {
                            String msg = "Filter not found for ARN:
%s".formatted(filterARN);
                            logger.warn(msg, rnfe);
                            throw new CompletionException(msg, rnfe);
                        }

                        throw new RuntimeException("Failed to delete the filter: " +
cause, cause);
                    }
                    return null;
                });
    }
```

```
   /**
    * Retrieves detailed information about a specific AWS Inspector2 finding
asynchronously.
    *
    * @param findingArn The ARN of the finding to look up.
    * @return A {@link CompletableFuture} that, when completed, provides a
formatted string
    * containing all available details for the finding.
    * @throws RuntimeException if the async call to Inspector2 fails.
    */
   public CompletableFuture<String> getFindingDetailsAsync(String findingArn) {
       BatchGetFindingDetailsRequest request =
BatchGetFindingDetailsRequest.builder()
               .findingArns(findingArn)
               .build();

       return getAsyncClient().batchGetFindingDetails(request)
               .thenApply(response -> {
                   if (response.findingDetails() == null ||
response.findingDetails().isEmpty()) {
                       return String.format("No details found for ARN: ",
findingArn);
                   }

                   StringBuilder sb = new StringBuilder();
                   response.findingDetails().forEach(detail -> {
                       sb.append("Finding ARN:
").append(detail.findingArn()).append("\n")
                                .append("Risk Score:
").append(detail.riskScore()).append("\n");

                       // ExploitObserved timings
                       if (detail.exploitObserved() != null) {
                           sb.append("Exploit First Seen:
").append(detail.exploitObserved().firstSeen()).append("\n")
                                    .append("Exploit Last Seen:
").append(detail.exploitObserved().lastSeen()).append("\n");
                       }

                       // Reference URLs
                       if (detail.hasReferenceUrls()) {
                           sb.append("Reference URLs:\n");
                           detail.referenceUrls().forEach(url -> sb.append("  •
").append(url).append("\n"));
```

```
                        }

                        // Tools
                        if (detail.hasTools()) {
                            sb.append("Tools:\n");
                            detail.tools().forEach(tool -> sb.append("  •
").append(tool).append("\n"));
                        }

                        // TTPs
                        if (detail.hasTtps()) {
                            sb.append("TTPs:\n");
                            detail.ttps().forEach(ttp -> sb.append("  •
").append(ttp).append("\n"));
                        }

                        // CWEs
                        if (detail.hasCwes()) {
                            sb.append("CWEs:\n");
                            detail.cwes().forEach(cwe -> sb.append("  •
").append(cwe).append("\n"));
                        }

                        // Evidence
                        if (detail.hasEvidences()) {
                            sb.append("Evidence:\n");
                            detail.evidences().forEach(ev -> {
                                sb.append("  - Severity:
").append(ev.severity()).append("\n");

                            });
                        }

                        sb.append("\n");
                    });

                    return sb.toString();
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ResourceNotFoundException rnfe) {
                        return "Finding not found: %s".formatted(findingArn);
                    }
```

```
                    // Fallback for other exceptions
                    throw new RuntimeException("Failed to get finding details for
 ARN: " + findingArn, cause);
                });
    }


    /**
     * Asynchronously disables AWS Inspector for the specified accounts and resource
 types.
     *
     * @param accountIds a {@link List} of AWS account IDs for which to disable
 Inspector;
     *                   may be {@code null} or empty to target the current account
     * @return a {@link CompletableFuture} that, when completed, returns a {@link
 String}
     *         summarizing the disable results for each account
     * @throws CompletionException if the disable operation fails due to validation
 errors,
     *                             service errors, or other exceptions
     * @see <a href="https://docs.aws.amazon.com/inspector/latest/APIReference/
API_Disable.html">
     *      AWS Inspector2 Disable API</a>
     */
    public CompletableFuture<String> disableInspectorAsync(List<String> accountIds)
 {

        // The resource types to disable.
        List<ResourceScanType> resourceTypes = List.of(
                ResourceScanType.EC2,
                ResourceScanType.ECR,
                ResourceScanType.LAMBDA,
                ResourceScanType.LAMBDA_CODE
        );

        // Build the request.
        DisableRequest.Builder requestBuilder = DisableRequest.builder()
                .resourceTypes(resourceTypes);

        if (accountIds != null && !accountIds.isEmpty()) {
            requestBuilder.accountIds(accountIds);
        }
```

```
        DisableRequest request = requestBuilder.build();

        return getAsyncClient().disable(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ValidationException) {
                            throw new CompletionException(
                                    "Inspector may already be disabled for this
account: %s".formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;
                            throw new CompletionException(
                                    "AWS Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to disable Inspector:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    StringBuilder summary = new StringBuilder("Disable results:\n");

                    if (response.accounts() == null ||
response.accounts().isEmpty()) {
                        summary.append("Inspector may already be disabled for all
target accounts.");
                        return summary.toString();
                    }

                    for (Account account : response.accounts()) {
                        String accountId = account.accountId() != null ?
account.accountId() : "Unknown";
```

```
                        String status = account.status() != null ?
 account.statusAsString() : "Unknown";
                        summary.append(" • Account: ").append(accountId)
                                .append(" # Status: ").append(status).append("\n");
                    }

                    return summary.toString();
                });
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - BatchGetAccountStatus
  - BatchGetFindingDetails
  - CreateFilter
  - DeleteFilter
  - 활성화
  - ListCoverage
  - ListCoverageStatistics
  - ListFilters
  - ListFindings
  - ListUsageTotals

작업

`BatchGetAccountStatus`

다음 코드 예시는 `BatchGetAccountStatus`의 사용 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Retrieves the account status using the Inspector2Client.
    */
   public CompletableFuture<String> getAccountStatusAsync() {
       BatchGetAccountStatusRequest request =
BatchGetAccountStatusRequest.builder()
               .accountIds(Collections.emptyList())
               .build();

       return getAsyncClient().batchGetAccountStatus(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof AccessDeniedException) {
                           throw new CompletionException(
                                   "You do not have sufficient access:
%s".formatted(cause.getMessage()),
                                   cause
                           );

                       }

                       if (cause instanceof Inspector2Exception) {
                           Inspector2Exception e = (Inspector2Exception) cause;

                           throw new CompletionException(
                                   "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                   e
                           );
                       }

                       throw new CompletionException(
                               "Unexpected error getting account status:
%s".formatted(exception.getMessage()),
                               exception
                       );
                   }
               })
               .thenApply(response -> {

                   StringBuilder sb = new StringBuilder();
```

```
                    List<AccountState> accounts = response.accounts();

                    if (accounts == null || accounts.isEmpty()) {
                        sb.append("No account status returned.\n");
                        return sb.toString();
                    }

                    sb.append("Inspector Account Status:\n");
                    for (AccountState account : accounts) {

                        String accountId = account.accountId() != null
                                ? account.accountId()
                                : "Unknown";

                        sb.append("  Account ID: ").append(accountId).append("\n");

                        // Overall account state
                        if (account.state() != null && account.state().status() !=
null) {
                            sb.append("  Overall State: ")
                                    .append(account.state().status())
                                    .append("\n");
                        } else {
                            sb.append("  Overall State: Unknown\n");
                        }

                        // Resource state (only status available)
                        ResourceState resources = account.resourceState();
                        if (resources != null) {
                            sb.append("  Resource Status: available\n");
                        }

                        sb.append("\n");
                    }

                    return sb.toString();
                });
    }
```

- API 세부 정보는 API 참조의 BatchGetAccountStatusAWS SDK for Java 2.x 를 참조하세요.

`BatchGetFindingDetails`

다음 코드 예시는 `BatchGetFindingDetails`의 사용 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Retrieves detailed information about a specific AWS Inspector2 finding
asynchronously.
    *
    * @param findingArn The ARN of the finding to look up.
    * @return A {@link CompletableFuture} that, when completed, provides a
formatted string
    * containing all available details for the finding.
    * @throws RuntimeException if the async call to Inspector2 fails.
    */
   public CompletableFuture<String> getFindingDetailsAsync(String findingArn) {
       BatchGetFindingDetailsRequest request =
BatchGetFindingDetailsRequest.builder()
               .findingArns(findingArn)
               .build();

       return getAsyncClient().batchGetFindingDetails(request)
               .thenApply(response -> {
                   if (response.findingDetails() == null ||
response.findingDetails().isEmpty()) {
                       return String.format("No details found for ARN: ",
findingArn);
                   }

                   StringBuilder sb = new StringBuilder();
                   response.findingDetails().forEach(detail -> {
                       sb.append("Finding ARN:
").append(detail.findingArn()).append("\n")
                               .append("Risk Score:
").append(detail.riskScore()).append("\n");
```

```
                        // ExploitObserved timings
                        if (detail.exploitObserved() != null) {
                            sb.append("Exploit First Seen:
").append(detail.exploitObserved().firstSeen()).append("\n")
                                    .append("Exploit Last Seen:
").append(detail.exploitObserved().lastSeen()).append("\n");
                        }

                        // Reference URLs
                        if (detail.hasReferenceUrls()) {
                            sb.append("Reference URLs:\n");
                            detail.referenceUrls().forEach(url -> sb.append("  •
").append(url).append("\n"));
                        }

                        // Tools
                        if (detail.hasTools()) {
                            sb.append("Tools:\n");
                            detail.tools().forEach(tool -> sb.append("  •
").append(tool).append("\n"));
                        }

                        // TTPs
                        if (detail.hasTtps()) {
                            sb.append("TTPs:\n");
                            detail.ttps().forEach(ttp -> sb.append("  •
").append(ttp).append("\n"));
                        }

                        // CWEs
                        if (detail.hasCwes()) {
                            sb.append("CWEs:\n");
                            detail.cwes().forEach(cwe -> sb.append("  •
").append(cwe).append("\n"));
                        }

                        // Evidence
                        if (detail.hasEvidences()) {
                            sb.append("Evidence:\n");
                            detail.evidences().forEach(ev -> {
                                sb.append("  - Severity:
").append(ev.severity()).append("\n");

                            });
```

```
                        }

                        sb.append("\n");
                    });

                    return sb.toString();
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ResourceNotFoundException rnfe) {
                        return "Finding not found: %s".formatted(findingArn);
                    }

                    // Fallback for other exceptions
                    throw new RuntimeException("Failed to get finding details for
ARN: " + findingArn, cause);
                });
    }
```

- API 세부 정보는 API 참조의 BatchGetFindingDetailsAWS SDK for Java 2.x 를 참조하세요.

`CreateFilter`

다음 코드 예시는 `CreateFilter`의 사용 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Creates a new LOW severity filter in AWS Inspector2 to suppress findings.
     *
     * @param filterName  the name of the filter to create
     * @param description a descriptive string explaining the purpose of the filter
     * @return a CompletableFuture that completes with the ARN of the created filter
```

```
     * @throws CompletionException wraps any validation, Inspector2 service, or
unexpected errors
     */
    public CompletableFuture<String> createLowSeverityFilterAsync(
            String filterName,
            String description) {

        // Define a filter to match LOW severity findings.
        StringFilter severityFilter = StringFilter.builder()
                .value(Severity.LOW.toString())
                .comparison(StringComparison.EQUALS)
                .build();

        // Create filter criteria.
        FilterCriteria filterCriteria = FilterCriteria.builder()
                .severity(Collections.singletonList(severityFilter))
                .build();

        // Build the filter creation request.
        CreateFilterRequest request = CreateFilterRequest.builder()
                .name(filterName)
                .filterCriteria(filterCriteria)
                .action(FilterAction.SUPPRESS)
                .description(description)
                .build();

        return getAsyncClient().createFilter(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ValidationException ve) {
                            throw new CompletionException(
                                    "Validation error creating filter:
%s".formatted(ve.getMessage()),
                                    ve
                            );
                        }

                        if (cause instanceof Inspector2Exception e) {
                            throw new CompletionException(
                                    "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
```

```
                                    e
                            );
                        }

                        // Unexpected async error
                        throw new CompletionException(
                                "Unexpected error creating filter:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                // Extract and return the ARN of the created filter.
                .thenApply(CreateFilterResponse::arn);
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateFilter를 참조하세요.

`DeleteFilter`

다음 코드 예시는 `DeleteFilter`의 사용 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Deletes an AWS Inspector2 filter.
     *
     * @param filterARN The ARN of the filter to delete.
     */
    public CompletableFuture<Void> deleteFilterAsync(String filterARN) {
        return getAsyncClient().deleteFilter(
                        DeleteFilterRequest.builder()
                                .arn(filterARN)
                                .build()
```

```
                )
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ResourceNotFoundException rnfe) {
                            String msg = "Filter not found for ARN:
%s".formatted(filterARN);
                            logger.warn(msg, rnfe);
                            throw new CompletionException(msg, rnfe);
                        }

                        throw new RuntimeException("Failed to delete the filter: " +
cause, cause);
                    }
                    return null;
                });
    }
```

- API 세부 정보는 API 참조의 DeleteFilterAWS SDK for Java 2.x 를 참조하세요.

`Disable`

다음 코드 예시는 `Disable`의 사용 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Asynchronously disables AWS Inspector for the specified accounts and resource
types.
     *
     * @param accountIds a {@link List} of AWS account IDs for which to disable
Inspector;
     *                   may be {@code null} or empty to target the current account
```

```
     * @return a {@link CompletableFuture} that, when completed, returns a {@link
 String}
     *         summarizing the disable results for each account
     * @throws CompletionException if the disable operation fails due to validation
 errors,
     *                             service errors, or other exceptions
     * @see <a href="https://docs.aws.amazon.com/inspector/latest/APIReference/
API_Disable.html">
     *      AWS Inspector2 Disable API</a>
     */
    public CompletableFuture<String> disableInspectorAsync(List<String> accountIds)
 {

        // The resource types to disable.
        List<ResourceScanType> resourceTypes = List.of(
                ResourceScanType.EC2,
                ResourceScanType.ECR,
                ResourceScanType.LAMBDA,
                ResourceScanType.LAMBDA_CODE
        );

        // Build the request.
        DisableRequest.Builder requestBuilder = DisableRequest.builder()
                .resourceTypes(resourceTypes);

        if (accountIds != null && !accountIds.isEmpty()) {
            requestBuilder.accountIds(accountIds);
        }

        DisableRequest request = requestBuilder.build();

        return getAsyncClient().disable(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ValidationException) {
                            throw new CompletionException(
                                    "Inspector may already be disabled for this
 account: %s".formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        if (cause instanceof Inspector2Exception) {
```

```
                            Inspector2Exception e = (Inspector2Exception) cause;
                            throw new CompletionException(
                                    "AWS Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to disable Inspector:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    StringBuilder summary = new StringBuilder("Disable results:\n");

                    if (response.accounts() == null ||
response.accounts().isEmpty()) {
                        summary.append("Inspector may already be disabled for all
target accounts.");
                        return summary.toString();
                    }

                    for (Account account : response.accounts()) {
                        String accountId = account.accountId() != null ?
account.accountId() : "Unknown";
                        String status = account.status() != null ?
account.statusAsString() : "Unknown";
                        summary.append(" • Account: ").append(accountId)
                                .append(" # Status: ").append(status).append("\n");
                    }

                    return summary.toString();
                });
    }
```

- API 세부 정보는 API 참조의 비활성화AWS SDK for Java 2.x 를 참조하세요.

`Enable`

다음 코드 예시는 `Enable`의 사용 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Enables AWS Inspector for the provided account(s) and default resource types.
 *
 * @param accountIds Optional list of AWS account IDs.
 */
public CompletableFuture<String> enableInspectorAsync(List<String> accountIds) {

    // The resource types to enable.
    List<ResourceScanType> resourceTypes = List.of(
            ResourceScanType.EC2,
            ResourceScanType.ECR,
            ResourceScanType.LAMBDA,
            ResourceScanType.LAMBDA_CODE
    );

    // Build the request.
    EnableRequest.Builder requestBuilder = EnableRequest.builder()
            .resourceTypes(resourceTypes);

    if (accountIds != null && !accountIds.isEmpty()) {
        requestBuilder.accountIds(accountIds);
    }

    EnableRequest request = requestBuilder.build();
    return getAsyncClient().enable(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ValidationException) {
                        throw new CompletionException(
```

```
                                    "Inspector may already be enabled for this
account: %s".formatted(cause.getMessage()),
                                    cause
                            );

                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;
                            throw new CompletionException(
                                    "AWS Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to enable Inspector:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    StringBuilder summary = new StringBuilder("Enable results:\n");

                    if (response.accounts() == null ||
response.accounts().isEmpty()) {
                        summary.append("Inspector may already be enabled for all
target accounts.");
                        return summary.toString();
                    }

                    for (Account account : response.accounts()) {
                        String accountId = account.accountId() != null ?
account.accountId() : "Unknown";
                        String status = account.status() != null ?
account.statusAsString() : "Unknown";
                        summary.append(" • Account: ").append(accountId)
                                .append(" # Status: ").append(status).append("\n");
                    }

                    return summary.toString();
                });
```

```
    }
```

- API 세부 정보는 API 참조의 활성화AWS SDK for Java 2.x 를 참조하세요.

`ListCoverage`

다음 코드 예시는 `ListCoverage`의 사용 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Lists AWS Inspector2 coverage details for scanned resources using a
paginator.
    *
    * @param maxResults Maximum number of resources to return.
    */
   public CompletableFuture<String> listCoverageAsync(int maxResults) {
       ListCoverageRequest initialRequest = ListCoverageRequest.builder()
               .maxResults(maxResults)
               .build();

       ListCoveragePublisher paginator =
getAsyncClient().listCoveragePaginator(initialRequest);
       StringBuilder summary = new StringBuilder();

       return paginator.subscribe(response -> {
           List<CoveredResource> coveredResources = response.coveredResources();

           if (coveredResources == null || coveredResources.isEmpty()) {
               summary.append("No coverage information available for this page.
\n");
               return;
           }
```

```
            Map<String, List<CoveredResource>> byType = coveredResources.stream()

.collect(Collectors.groupingBy(CoveredResource::resourceTypeAsString));

            byType.forEach((type, list) ->
                    summary.append("  ").append(type)
                            .append(": ").append(list.size())
                            .append(" resource(s)\n")
            );

            // Include up to 3 sample resources per page
            for (int i = 0; i < Math.min(coveredResources.size(), 3); i++) {
                CoveredResource r = coveredResources.get(i);
                summary.append("  - ").append(r.resourceTypeAsString())
                        .append(": ").append(r.resourceId()).append("\n");
                summary.append("    Scan Type:
").append(r.scanTypeAsString()).append("\n");
                if (r.scanStatus() != null) {
                    summary.append("    Status:
").append(r.scanStatus().statusCodeAsString()).append("\n");
                }
                if (r.accountId() != null) {
                    summary.append("    Account ID:
").append(r.accountId()).append("\n");
                }
                summary.append("\n");
            }

        }).thenApply(v -> {
            if (summary.length() == 0) {
                return "No coverage information found across all pages.";
            } else {
                return "Coverage Information:\n" + summary.toString();
            }
        }).exceptionally(ex -> {
            Throwable cause = ex.getCause();
            if (cause instanceof ValidationException) {
                throw new CompletionException(
                        "Validation error listing coverage: " + cause.getMessage(),
cause);
            } else if (cause instanceof Inspector2Exception e) {
                throw new CompletionException(
                        "Inspector2 service error: " +
e.awsErrorDetails().errorMessage(), e);
```

```
            }
            throw new CompletionException("Unexpected error listing coverage: " +
ex.getMessage(), ex);
        });
    }
```

- API 세부 정보는 API 참조의 ListCoverageAWS SDK for Java 2.x 를 참조하세요.

`ListCoverageStatistics`

다음 코드 예시는 `ListCoverageStatistics`의 사용 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Retrieves and prints the coverage statistics using a paginator.
     */
    public CompletableFuture<String> listCoverageStatisticsAsync() {
        ListCoverageStatisticsRequest request =
ListCoverageStatisticsRequest.builder()
                .build();

        return getAsyncClient().listCoverageStatistics(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();

                        if (cause instanceof ValidationException) {
                            throw new CompletionException(
                                    "Validation error listing coverage statistics:
%s".formatted(cause.getMessage()),
                                    cause
                            );
```

```
                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;

                            throw new CompletionException(
                                    "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    e
                            );
                        }

                        throw new CompletionException(
                                "Unexpected error listing coverage statistics:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    List<Counts> countsList = response.countsByGroup();
                    StringBuilder sb = new StringBuilder();

                    if (countsList == null || countsList.isEmpty()) {
                        sb.append("No coverage statistics available.\n");
                        return sb.toString();
                    }

                    sb.append("Coverage Statistics:\n");

                    for (Counts c : countsList) {
                        sb.append("  Group: ").append(c.groupKey()).append("\n")
                                .append("    Total Count:
").append(c.count()).append("\n\n");
                    }

                    return sb.toString();
                });
    }
```

- API 세부 정보는 API 참조의 ListCoverageStatisticsAWS SDK for Java 2.x 를 참조하세요.

`ListFilters`

다음 코드 예시는 `ListFilters`의 사용 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Asynchronously lists Inspector2 filters using a paginator.
     *
     * @param maxResults maximum filters per page (nullable)
     * @return CompletableFuture completed with summary text
     */
    public CompletableFuture<String> listFiltersAsync(Integer maxResults) {
        logger.info("Starting async filters paginator…");

        ListFiltersRequest.Builder builder = ListFiltersRequest.builder();
        if (maxResults != null) {
            builder.maxResults(maxResults);
        }

        ListFiltersRequest request = builder.build();

        // Paginator from SDK
        ListFiltersPublisher paginator =
getAsyncClient().listFiltersPaginator(request);
        StringBuilder collectedFilterIds = new StringBuilder();

        return paginator.subscribe(response -> {
            response.filters().forEach(filter -> {
                logger.info("Filter: " + filter.arn());
                collectedFilterIds.append(filter.arn()).append("\n");
            });
        }).thenApply(v -> {
            String result = collectedFilterIds.toString();
            logger.info("Successfully listed all filters.");
            return result.isEmpty() ? "No filters found." : result;
```

```
        }).exceptionally(ex -> {
            Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

            if (cause instanceof ValidationException ve) {
                throw new CompletionException(
                        "Validation error listing filters:
%s".formatted(ve.getMessage()),
                        ve
                );
            }

            throw new RuntimeException("Failed to list filters", ex);
        });
    }
```

- API 세부 정보는 API 참조의 ListFiltersAWS SDK for Java 2.x 를 참조하세요.

`ListFindings`

다음 코드 예시는 `ListFindings`의 사용 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Lists all AWS Inspector findings of LOW severity asynchronously.
     *
     * @return CompletableFuture containing a List of finding ARNs.
     * Returns an empty list if no LOW severity findings are found.
     */
    public CompletableFuture<ArrayList<String>> listLowSeverityFindingsAsync() {
        logger.info("Starting async LOW severity findings paginator…");

        // Build a filter criteria for LOW severity.
        StringFilter severityFilter = StringFilter.builder()
```

```
                .value(Severity.LOW.toString())
                .comparison(StringComparison.EQUALS)
                .build();

        FilterCriteria filterCriteria = FilterCriteria.builder()
                .severity(Collections.singletonList(severityFilter))
                .build();

        // Build the request.
        ListFindingsRequest request = ListFindingsRequest.builder()
                .filterCriteria(filterCriteria)
                .build();

        ListFindingsPublisher paginator =
 getAsyncClient().listFindingsPaginator(request);
        List<String> allArns = Collections.synchronizedList(new ArrayList<>());

        return paginator.subscribe(response -> {
                    if (response.findings() != null && !
response.findings().isEmpty()) {
                        response.findings().forEach(finding -> {
                            logger.info("Finding ARN: {}", finding.findingArn());
                            allArns.add(finding.findingArn());
                        });
                    } else {
                        logger.info("Page contained no findings.");
                    }
                })
                .thenRun(() -> logger.info("Successfully listed all LOW severity
 findings."))
                .thenApply(v -> new ArrayList<>(allArns)) // Return list instead of
 a formatted string
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;
                    if (cause instanceof ValidationException ve) {
                        throw new CompletionException(
                                "Validation error listing LOW severity findings:
 %s".formatted(ve.getMessage()),
                                ve
                        );
                    }
                    throw new RuntimeException("Failed to list LOW severity
 findings", ex);
                });
```

```
    }
```

- API 세부 정보는 API 참조의 ListFindingsAWS SDK for Java 2.x 를 참조하세요.

`ListUsageTotals`

다음 코드 예시는 `ListUsageTotals`의 사용 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Asynchronously lists Inspector2 usage totals using a paginator.
     *
     * @param accountIds optional list of account IDs
     * @param maxResults maximum results per page
     * @return CompletableFuture completed with formatted summary text
     */
    public CompletableFuture<String> listUsageTotalsAsync(
            List<String> accountIds,
            int maxResults) {

        logger.info("Starting usage totals paginator…");

        ListUsageTotalsRequest.Builder builder = ListUsageTotalsRequest.builder()
                .maxResults(maxResults);

        if (accountIds != null && !accountIds.isEmpty()) {
            builder.accountIds(accountIds);
        }

        ListUsageTotalsRequest request = builder.build();
        ListUsageTotalsPublisher paginator =
getAsyncClient().listUsageTotalsPaginator(request);
```

```
        StringBuilder summaryBuilder = new StringBuilder();

        return paginator.subscribe(response -> {
                    if (response.totals() != null && !response.totals().isEmpty()) {
                        response.totals().forEach(total -> {
                            if (total.usage() != null) {
                                total.usage().forEach(usage -> {
                                    logger.info("Usage: {} = {}",
usage.typeAsString(), usage.total());
                                    summaryBuilder.append(usage.typeAsString())
                                            .append(": ")
                                            .append(usage.total())
                                            .append("\n");
                                });
                            }
                        });
                    } else {
                        logger.info("Page contained no usage totals.");
                    }
                }).thenRun(() -> logger.info("Successfully listed usage totals."))
                .thenApply(v -> {
                    String summary = summaryBuilder.toString();
                    return summary.isEmpty() ? "No usage totals found." : summary;
                }).exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ValidationException ve) {
                        throw new CompletionException(
                                "Validation error listing usage totals:
%s".formatted(ve.getMessage()),
                                ve
                        );
                    }

                    throw new CompletionException("Failed to list usage totals",
cause);
                });
    }
```

- API 세부 정보는 API 참조의 AWS SDK for Java 2.x ListUsageTotals.

AWS IoT SDK for Java 2.x를 사용한 예제

다음 코드 예제에서는를와 AWS SDK for Java 2.x 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS IoT.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시작하기

안녕하세요 AWS IoT

다음 코드 예제에서는 AWS IoT를 사용하여 시작하는 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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 software.amazon.awssdk.services.iot.paginators.ListThingsIterable;
```

```
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) {
        iotClient.listThingsPaginator(ListThingsRequest.builder()
                .maxResults(10)
                .build())
            .stream()
            .flatMap(response -> response.things().stream())
            .forEach(attribute -> {
                System.out.println("Thing name: " + attribute.thingName());
                System.out.println("Thing ARN: " + attribute.thingArn());
            });
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 listThings를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- AWS IoT 사물을 생성합니다.
- 디바이스 인증서를 생성합니다.
- 속성을 사용하여 AWS IoT 사물을 업데이트합니다.
- 고유한 엔드포인트를 반환합니다.
- AWS IoT 인증서를 나열합니다.

- 섀도우를 업데이트 AWS IoT 합니다.
- 상태 정보를 씁니다.
- 규칙을 생성합니다.
- 규칙을 나열합니다.
- 사물 이름을 사용하여 사물을 검색합니다.
- AWS IoT 사물을 삭제합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

AWS IoT 기능을 보여주는 대화형 시나리오를 실행합니다.

```
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 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", "-");

    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);
        }

        IotActions iotActions = new IotActions();
        String thingName;
        String ruleName;
        String roleARN = args[0];
        String snsAction = args[1];
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS IoT basics scenario.");
        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.

            Let's get started...
```

```
            """);
        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();
        iotActions.createIoTThing(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 = iotActions.createCertificate();
            System.out.println("Attach the certificate to the AWS IoT Thing.");
            iotActions.attachCertificateToThing(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.
            """);
```

```
        waitForInputToContinue(scanner);
        iotActions.updateShadowThing(thingName);
        waitForInputToContinue(scanner);
        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.
           """);
        waitForInputToContinue(scanner);
        String endpointUrl = iotActions.describeEndpoint();
        System.out.println("The endpoint is "+endpointUrl);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. List your AWS IoT certificates");
        waitForInputToContinue(scanner);
        if (certificateArn.length() > 0) {
            iotActions.listCertificates();
        } else {
            System.out.println("You did not create a certificates. Skipping this
step.");
        }
        waitForInputToContinue(scanner);
        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.
           """);
        waitForInputToContinue(scanner);
        iotActions.updateShadowThing(thingName);
```

```
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Write out the state information, in JSON format.");
        waitForInputToContinue(scanner);
        iotActions.getPayload(thingName);
        waitForInputToContinue(scanner);
        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();
        iotActions.createIoTRule(roleARN, ruleName, snsAction);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. List your rules.");
        waitForInputToContinue(scanner);
        iotActions.listIoTRules();
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Search things using the Thing name.");
        waitForInputToContinue(scanner);
        String queryString = "thingName:"+thingName ;
        iotActions.searchThings(queryString);
        waitForInputToContinue(scanner);
        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.");
                waitForInputToContinue(scanner);
                iotActions.detachThingPrincipal(thingName, certificateArn);
                iotActions.deleteCertificate(certificateArn);
                waitForInputToContinue(scanner);
            } 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")) {
            iotActions.deleteIoTThing(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);
    }


    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.");
            }
        }
    }
}
```

AWS IoT SDK 메서드의 래퍼 클래스입니다.

```
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
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.iot.IotAsyncClient;
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.Certificate;
import software.amazon.awssdk.services.iot.model.CreateKeysAndCertificateResponse;
import software.amazon.awssdk.services.iot.model.CreateThingRequest;
import software.amazon.awssdk.services.iot.model.CreateThingResponse;
import software.amazon.awssdk.services.iot.model.CreateTopicRuleRequest;
import software.amazon.awssdk.services.iot.model.CreateTopicRuleResponse;
import software.amazon.awssdk.services.iot.model.DeleteCertificateRequest;
import software.amazon.awssdk.services.iot.model.DeleteCertificateResponse;
import software.amazon.awssdk.services.iot.model.DeleteThingRequest;
import software.amazon.awssdk.services.iot.model.DeleteThingResponse;
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.DetachThingPrincipalResponse;
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.TopicRuleListItem;
import software.amazon.awssdk.services.iot.model.SnsAction;
import software.amazon.awssdk.services.iot.model.TopicRulePayload;
import software.amazon.awssdk.services.iotdataplane.IotDataPlaneAsyncClient;
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 software.amazon.awssdk.services.iotdataplane.model.UpdateThingShadowResponse;
import java.nio.charset.StandardCharsets;
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class IotActions {

    private static IotAsyncClient iotAsyncClient;

    private static IotDataPlaneAsyncClient iotAsyncDataPlaneClient;

    private static final String TOPIC = "your-iot-topic";

    private static IotDataPlaneAsyncClient getAsyncDataPlaneClient() {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
            .connectionTimeout(Duration.ofSeconds(60))
            .readTimeout(Duration.ofSeconds(60))
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))
            .apiCallAttemptTimeout(Duration.ofSeconds(90))
            .retryPolicy(RetryPolicy.builder()
                .numRetries(3)
                .build())
            .build();

        if (iotAsyncDataPlaneClient == null) {
            iotAsyncDataPlaneClient = IotDataPlaneAsyncClient.builder()
```

```
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return iotAsyncDataPlaneClient;
    }


    private static IotAsyncClient getAsyncClient() {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
            .connectionTimeout(Duration.ofSeconds(60))
            .readTimeout(Duration.ofSeconds(60))
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))
            .apiCallAttemptTimeout(Duration.ofSeconds(90))
            .retryPolicy(RetryPolicy.builder()
                .numRetries(3)
                .build())
            .build();

        if (iotAsyncClient == null) {
            iotAsyncClient = IotAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return iotAsyncClient;
    }

    /**
     * Creates an IoT certificate asynchronously.
     *
     * @return The ARN of the created certificate.
     * <p>
     * This method initiates an asynchronous request to create an IoT certificate.
     * If the request is successful, it prints the certificate details and returns
the certificate ARN.
```

```
     * If an exception occurs, it prints the error message.
     */
    public String createCertificate() {
        CompletableFuture<CreateKeysAndCertificateResponse> future =
getAsyncClient().createKeysAndCertificate();
        final String[] certificateArn = {null};
        future.whenComplete((response, ex) -> {
            if (response != null) {
                String certificatePem = response.certificatePem();
                certificateArn[0] = response.certificateArn();

                // Print the details.
                System.out.println("\nCertificate:");
                System.out.println(certificatePem);
                System.out.println("\nCertificate ARN:");
                System.out.println(certificateArn[0]);

            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + cause.getMessage());
                }
            }
        });

        future.join();
        return certificateArn[0];
    }

    /**
     * Creates an IoT Thing with the specified name asynchronously.
     *
     * @param thingName The name of the IoT Thing to create.
     *
     * This method initiates an asynchronous request to create an IoT Thing with the
specified name.
     * If the request is successful, it prints the name of the thing and its ARN
value.
     * If an exception occurs, it prints the error message.
     */
```

```
   public void createIoTThing(String thingName) {
       CreateThingRequest createThingRequest = CreateThingRequest.builder()
           .thingName(thingName)
           .build();

       CompletableFuture<CreateThingResponse> future =
getAsyncClient().createThing(createThingRequest);
       future.whenComplete((createThingResponse, ex) -> {
           if (createThingResponse != null) {
               System.out.println(thingName + " was successfully created. The ARN
value is " + createThingResponse.thingArn());
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else {
                   System.err.println("Unexpected error: " + cause.getMessage());
               }
           }
       });

       future.join();
   }

   /**
    * Attaches a certificate to an IoT Thing asynchronously.
    *
    * @param thingName The name of the IoT Thing.
    * @param certificateArn The ARN of the certificate to attach.
    *
    * This method initiates an asynchronous request to attach a certificate to an
IoT Thing.
    * If the request is successful, it prints a confirmation message and additional
information about the Thing.
    * If an exception occurs, it prints the error message.
    */
   public void attachCertificateToThing(String thingName, String certificateArn) {
       AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
           .thingName(thingName)
           .principal(certificateArn)
           .build();
```

```
        CompletableFuture<AttachThingPrincipalResponse> future =
getAsyncClient().attachThingPrincipal(principalRequest);
        future.whenComplete((attachResponse, ex) -> {
            if (attachResponse != null &&
attachResponse.sdkHttpResponse().isSuccessful()) {
                System.out.println("Certificate attached to Thing successfully.");

                // Print additional information about the Thing.
                describeThing(thingName);
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to attach certificate to Thing. HTTP
Status Code: " +
                            attachResponse.sdkHttpResponse().statusCode());
                }
            }
        });

        future.join();
    }

    /**
     * Describes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to describe an IoT Thing.
     * If the request is successful, it prints the Thing details.
     * If an exception occurs, it prints the error message.
     */
    private void describeThing(String thingName) {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DescribeThingResponse> future =
getAsyncClient().describeThing(thingRequest);
        future.whenComplete((describeResponse, ex) -> {
```

```
            if (describeResponse != null) {
                System.out.println("Thing Details:");
                System.out.println("Thing Name: " + describeResponse.thingName());
                System.out.println("Thing ARN: " + describeResponse.thingArn());
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to describe Thing.");
                }
            }
        });

        future.join();
    }

    /**
     * Updates the shadow of an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to update the shadow of an IoT
Thing.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void updateShadowThing(String thingName) {
        // 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();

        CompletableFuture<UpdateThingShadowResponse> future =
getAsyncDataPlaneClient().updateThingShadow(updateThingShadowRequest);
        future.whenComplete((updateResponse, ex) -> {
```

```
            if (updateResponse != null) {
                System.out.println("Thing Shadow updated successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to update Thing Shadow.");
                }
            }
        });

        future.join();
    }

    /**
     * Describes the endpoint of the IoT service asynchronously.
     *
     * @return A CompletableFuture containing the full endpoint URL.
     *
     * This method initiates an asynchronous request to describe the endpoint of the
 IoT service.
     * If the request is successful, it prints and returns the full endpoint URL.
     * If an exception occurs, it prints the error message.
     */
    public String describeEndpoint() {
        CompletableFuture<DescribeEndpointResponse> future =
 getAsyncClient().describeEndpoint(DescribeEndpointRequest.builder().endpointType("iot:Data-
ATS").build());
        final String[] result = {null};

        future.whenComplete((endpointResponse, ex) -> {
            if (endpointResponse != null) {
                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);
                result[0] = fullEndpoint;
            } else {
```

```
                Throwable cause = (ex instanceof CompletionException) ?
 ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + cause.getMessage());
                }
            }
        });

        future.join();
        return result[0];
    }

    /**
     * Extracts a specific value from the endpoint URL.
     *
     * @param input The endpoint URL to process.
     * @return The extracted value from the endpoint URL.
     */
    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 "" ;
    }

    /**
     * Lists all certificates asynchronously.
     *
```

```
    * This method initiates an asynchronous request to list all certificates.
    * If the request is successful, it prints the certificate IDs and ARNs.
    * If an exception occurs, it prints the error message.
    */
   public void listCertificates() {
       CompletableFuture<ListCertificatesResponse> future =
getAsyncClient().listCertificates();
       future.whenComplete((response, ex) -> {
           if (response != null) {
               List<Certificate> certList = response.certificates();
               for (Certificate cert : certList) {
                   System.out.println("Cert id: " + cert.certificateId());
                   System.out.println("Cert Arn: " + cert.certificateArn());
               }
           } else {
               Throwable cause = ex != null ? ex.getCause() : null;
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else if (cause != null) {
                   System.err.println("Unexpected error: " + cause.getMessage());
               } else {
                   System.err.println("Failed to list certificates.");
               }
           }
       });

       future.join();
   }

   /**
    * Retrieves the payload of a Thing's shadow asynchronously.
    *
    * @param thingName The name of the IoT Thing.
    *
    * This method initiates an asynchronous request to get the payload of a Thing's
shadow.
    * If the request is successful, it prints the shadow data.
    * If an exception occurs, it prints the error message.
    */
   public void getPayload(String thingName) {
       GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
           .thingName(thingName)
```

```
            .build();

        CompletableFuture<GetThingShadowResponse> future =
getAsyncDataPlaneClient().getThingShadow(getThingShadowRequest);
        future.whenComplete((getThingShadowResponse, ex) -> {
            if (getThingShadowResponse != null) {
                // Extracting payload from response.
                SdkBytes payload = getThingShadowResponse.payload();
                String payloadString = payload.asUtf8String();
                System.out.println("Received Shadow Data: " + payloadString);
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to get Thing Shadow payload.");
                }
            }
        });

        future.join();
    }

    /**
     * Creates an IoT rule asynchronously.
     *
     * @param roleARN The ARN of the IAM role that grants access to the rule's
actions.
     * @param ruleName The name of the IoT rule.
     * @param action The ARN of the action to perform when the rule is triggered.
     *
     * This method initiates an asynchronous request to create an IoT rule.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void createIoTRule(String roleARN, String ruleName, String action) {
        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();

        CompletableFuture<CreateTopicRuleResponse> future =
getAsyncClient().createTopicRule(topicRuleRequest);
        future.whenComplete((response, ex) -> {
            if (response != null) {
                System.out.println("IoT Rule created successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to create IoT Rule.");
                }
            }
        });

        future.join();
    }

    /**
     * Lists IoT rules asynchronously.
     *
     * This method initiates an asynchronous request to list IoT rules.
     * If the request is successful, it prints the names and ARNs of the rules.
```

```
     * If an exception occurs, it prints the error message.
     */
    public void listIoTRules() {
        ListTopicRulesRequest listTopicRulesRequest =
ListTopicRulesRequest.builder().build();
        CompletableFuture<ListTopicRulesResponse> future =
getAsyncClient().listTopicRules(listTopicRulesRequest);
        future.whenComplete((listTopicRulesResponse, ex) -> {
            if (listTopicRulesResponse != null) {
                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("--------------");
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to list IoT Rules.");
                }
            }
        });

        future.join();
    }

    /**
     * Searches for IoT Things asynchronously based on a query string.
     *
     * @param queryString The query string to search for Things.
     *
     * This method initiates an asynchronous request to search for IoT Things.
     * If the request is successful and Things are found, it prints their IDs.
     * If no Things are found, it prints a message indicating so.
     * If an exception occurs, it prints the error message.
     */
    public void searchThings(String queryString) {
        SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
```

```
            .queryString(queryString)
            .build();

        CompletableFuture<SearchIndexResponse> future =
getAsyncClient().searchIndex(searchIndexRequest);
        future.whenComplete((searchIndexResponse, ex) -> {
            if (searchIndexResponse != null) {
                // 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()));
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to search for IoT Things.");
                }
            }
        });

        future.join();
    }

    /**
     * Detaches a principal (certificate) from an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     * @param certificateArn The ARN of the certificate to detach.
     *
     * This method initiates an asynchronous request to detach a certificate from an
IoT Thing.
     * If the detachment is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void detachThingPrincipal(String thingName, String certificateArn) {
        DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
```

```
            .principal(certificateArn)
            .thingName(thingName)
            .build();

        CompletableFuture<DetachThingPrincipalResponse> future =
getAsyncClient().detachThingPrincipal(thingPrincipalRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println(certificateArn + " was successfully removed from
" + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }

    /**
     * Deletes a certificate asynchronously.
     *
     * @param certificateArn The ARN of the certificate to delete.
     *
     * This method initiates an asynchronous request to delete a certificate.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteCertificate(String certificateArn) {
        DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
            .certificateId(extractCertificateId(certificateArn))
            .build();

        CompletableFuture<DeleteCertificateResponse> future =
getAsyncClient().deleteCertificate(certificateProviderRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println(certificateArn + " was successfully deleted.");
```

```
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }

    /**
     * Deletes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing to delete.
     *
     * This method initiates an asynchronous request to delete an IoT Thing.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteIoTThing(String thingName) {
        DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DeleteThingResponse> future =
getAsyncClient().deleteThing(deleteThingRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println("Deleted Thing " + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });
```

```
        future.join();
    }

    // Get the cert Id  from the Cert ARN value.
    private 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);
    }
}
```

작업

`AttachThingPrincipal`

다음 코드 예시는 `AttachThingPrincipal`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Attaches a certificate to an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     * @param certificateArn The ARN of the certificate to attach.
     *
     * This method initiates an asynchronous request to attach a certificate to an
 IoT Thing.
     * If the request is successful, it prints a confirmation message and additional
 information about the Thing.
     * If an exception occurs, it prints the error message.
     */
    public void attachCertificateToThing(String thingName, String certificateArn) {
```

```
        AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
            .thingName(thingName)
            .principal(certificateArn)
            .build();

        CompletableFuture<AttachThingPrincipalResponse> future =
getAsyncClient().attachThingPrincipal(principalRequest);
        future.whenComplete((attachResponse, ex) -> {
            if (attachResponse != null &&
attachResponse.sdkHttpResponse().isSuccessful()) {
                System.out.println("Certificate attached to Thing successfully.");

                // Print additional information about the Thing.
                describeThing(thingName);
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to attach certificate to Thing. HTTP
Status Code: " +
                            attachResponse.sdkHttpResponse().statusCode());
                }
            }
        });

        future.join();
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 AttachThingPrincipal을 참조하세요.

`CreateKeysAndCertificate`

다음 코드 예시는 `CreateKeysAndCertificate`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Creates an IoT certificate asynchronously.
    *
    * @return The ARN of the created certificate.
    * <p>
    * This method initiates an asynchronous request to create an IoT certificate.
    * If the request is successful, it prints the certificate details and returns
the certificate ARN.
    * If an exception occurs, it prints the error message.
    */
   public String createCertificate() {
       CompletableFuture<CreateKeysAndCertificateResponse> future =
getAsyncClient().createKeysAndCertificate();
       final String[] certificateArn = {null};
       future.whenComplete((response, ex) -> {
           if (response != null) {
               String certificatePem = response.certificatePem();
               certificateArn[0] = response.certificateArn();

               // Print the details.
               System.out.println("\nCertificate:");
               System.out.println(certificatePem);
               System.out.println("\nCertificate ARN:");
               System.out.println(certificateArn[0]);

           } else {
               Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
               } else {
                   System.err.println("Unexpected error: " + cause.getMessage());
               }
```

```
            }
        });

        future.join();
        return certificateArn[0];
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조에서 CreateKeysAndCertificate를 참조하세요.

`CreateThing`

다음 코드 예시는 `CreateThing`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Creates an IoT Thing with the specified name asynchronously.
     *
     * @param thingName The name of the IoT Thing to create.
     *
     * This method initiates an asynchronous request to create an IoT Thing with the
specified name.
     * If the request is successful, it prints the name of the thing and its ARN
value.
     * If an exception occurs, it prints the error message.
     */
    public void createIoTThing(String thingName) {
        CreateThingRequest createThingRequest = CreateThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<CreateThingResponse> future =
getAsyncClient().createThing(createThingRequest);
        future.whenComplete((createThingResponse, ex) -> {
            if (createThingResponse != null) {
```

```
                System.out.println(thingName + " was successfully created. The ARN
value is " + createThingResponse.thingArn());
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + cause.getMessage());
                }
            }
        });

        future.join();
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateThing을 참조하세요.

`CreateTopicRule`

다음 코드 예시는 `CreateTopicRule`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Creates an IoT rule asynchronously.
     *
     * @param roleARN The ARN of the IAM role that grants access to the rule's
actions.
     * @param ruleName The name of the IoT rule.
     * @param action The ARN of the action to perform when the rule is triggered.
     *
     * This method initiates an asynchronous request to create an IoT rule.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
```

```
     */
    public void createIoTRule(String roleARN, String ruleName, String action) {
        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();

        CompletableFuture<CreateTopicRuleResponse> future =
getAsyncClient().createTopicRule(topicRuleRequest);
        future.whenComplete((response, ex) -> {
            if (response != null) {
                System.out.println("IoT Rule created successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to create IoT Rule.");
                }
            }
        });

        future.join();
```

```
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateTopicRule을 참조하세요.

`DeleteCertificate`

다음 코드 예시는 `DeleteCertificate`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Deletes a certificate asynchronously.
    *
    * @param certificateArn The ARN of the certificate to delete.
    *
    * This method initiates an asynchronous request to delete a certificate.
    * If the deletion is successful, it prints a confirmation message.
    * If an exception occurs, it prints the error message.
    */
   public void deleteCertificate(String certificateArn) {
       DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
           .certificateId(extractCertificateId(certificateArn))
           .build();

       CompletableFuture<DeleteCertificateResponse> future =
getAsyncClient().deleteCertificate(certificateProviderRequest);
       future.whenComplete((voidResult, ex) -> {
           if (ex == null) {
               System.out.println(certificateArn + " was successfully deleted.");
           } else {
               Throwable cause = ex.getCause();
               if (cause instanceof IotException) {
                   System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
```

```
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteCertificate를 참조하세요.

`DeleteThing`

다음 코드 예시는 `DeleteThing`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Deletes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing to delete.
     *
     * This method initiates an asynchronous request to delete an IoT Thing.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteIoTThing(String thingName) {
        DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DeleteThingResponse> future =
getAsyncClient().deleteThing(deleteThingRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
```

```
                System.out.println("Deleted Thing " + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteThing을 참조하세요.

`DescribeEndpoint`

다음 코드 예시는 `DescribeEndpoint`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Describes the endpoint of the IoT service asynchronously.
     *
     * @return A CompletableFuture containing the full endpoint URL.
     *
     * This method initiates an asynchronous request to describe the endpoint of the
IoT service.
     * If the request is successful, it prints and returns the full endpoint URL.
     * If an exception occurs, it prints the error message.
     */
    public String describeEndpoint() {
```

```
        CompletableFuture<DescribeEndpointResponse> future =
 getAsyncClient().describeEndpoint(DescribeEndpointRequest.builder().endpointType("iot:Data-
ATS").build());
        final String[] result = {null};

        future.whenComplete((endpointResponse, ex) -> {
            if (endpointResponse != null) {
                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);
                result[0] = fullEndpoint;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
 ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + cause.getMessage());
                }
            }
        });

        future.join();
        return result[0];
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeEndpoint를 참조하세요.

`DescribeThing`

다음 코드 예시는 `DescribeThing`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Describes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to describe an IoT Thing.
     * If the request is successful, it prints the Thing details.
     * If an exception occurs, it prints the error message.
     */
    private void describeThing(String thingName) {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DescribeThingResponse> future =
getAsyncClient().describeThing(thingRequest);
        future.whenComplete((describeResponse, ex) -> {
            if (describeResponse != null) {
                System.out.println("Thing Details:");
                System.out.println("Thing Name: " + describeResponse.thingName());
                System.out.println("Thing ARN: " + describeResponse.thingArn());
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to describe Thing.");
                }
            }
        });
```

```
        future.join();
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeThing을 참조하세요.

`DetachThingPrincipal`

다음 코드 예시는 `DetachThingPrincipal`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Detaches a principal (certificate) from an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     * @param certificateArn The ARN of the certificate to detach.
     *
     * This method initiates an asynchronous request to detach a certificate from an
IoT Thing.
     * If the detachment is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void detachThingPrincipal(String thingName, String certificateArn) {
        DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
            .principal(certificateArn)
            .thingName(thingName)
            .build();

        CompletableFuture<DetachThingPrincipalResponse> future =
getAsyncClient().detachThingPrincipal(thingPrincipalRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println(certificateArn + " was successfully removed from
" + thingName);
```

```
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DetachThingPrincipal을 참조하세요.

`ListCertificates`

다음 코드 예시는 `ListCertificates`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Lists all certificates asynchronously.
     *
     * This method initiates an asynchronous request to list all certificates.
     * If the request is successful, it prints the certificate IDs and ARNs.
     * If an exception occurs, it prints the error message.
     */
    public void listCertificates() {
        CompletableFuture<ListCertificatesResponse> future =
getAsyncClient().listCertificates();
        future.whenComplete((response, ex) -> {
            if (response != null) {
                List<Certificate> certList = response.certificates();
```

```
                for (Certificate cert : certList) {
                    System.out.println("Cert id: " + cert.certificateId());
                    System.out.println("Cert Arn: " + cert.certificateArn());
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to list certificates.");
                }
            }
        });

        future.join();
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListCertificates를 참조하세요.

`SearchIndex`

다음 코드 예시는 `SearchIndex`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Searches for IoT Things asynchronously based on a query string.
     *
     * @param queryString The query string to search for Things.
     *
     * This method initiates an asynchronous request to search for IoT Things.
     * If the request is successful and Things are found, it prints their IDs.
```

```
     * If no Things are found, it prints a message indicating so.
     * If an exception occurs, it prints the error message.
     */
    public void searchThings(String queryString) {
        SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
            .queryString(queryString)
            .build();

        CompletableFuture<SearchIndexResponse> future =
getAsyncClient().searchIndex(searchIndexRequest);
        future.whenComplete((searchIndexResponse, ex) -> {
            if (searchIndexResponse != null) {
                // 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()));
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to search for IoT Things.");
                }
            }
        });

        future.join();
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 SearchIndex를 참조하세요.

AWS IoT data SDK for Java 2.x를 사용한 예제

다음 코드 예제에서는를와 AWS SDK for Java 2.x 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS IoT data.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`GetThingShadow`

다음 코드 예시는 `GetThingShadow`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Retrieves the payload of a Thing's shadow asynchronously.
    *
    * @param thingName The name of the IoT Thing.
    *
    * This method initiates an asynchronous request to get the payload of a Thing's
shadow.
    * If the request is successful, it prints the shadow data.
    * If an exception occurs, it prints the error message.
    */
   public void getPayload(String thingName) {
       GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
           .thingName(thingName)
           .build();
```

```
        CompletableFuture<GetThingShadowResponse> future =
getAsyncDataPlaneClient().getThingShadow(getThingShadowRequest);
        future.whenComplete((getThingShadowResponse, ex) -> {
            if (getThingShadowResponse != null) {
                // Extracting payload from response.
                SdkBytes payload = getThingShadowResponse.payload();
                String payloadString = payload.asUtf8String();
                System.out.println("Received Shadow Data: " + payloadString);
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to get Thing Shadow payload.");
                }
            }
        });

        future.join();
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetThingShadow를 참조하세요.

`UpdateThingShadow`

다음 코드 예시는 `UpdateThingShadow`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Updates the shadow of an IoT Thing asynchronously.
```

```
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to update the shadow of an IoT
Thing.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void updateShadowThing(String thingName) {
        // 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();

        CompletableFuture<UpdateThingShadowResponse> future =
getAsyncDataPlaneClient().updateThingShadow(updateThingShadowRequest);
        future.whenComplete((updateResponse, ex) -> {
            if (updateResponse != null) {
                System.out.println("Thing Shadow updated successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to update Thing Shadow.");
                }
            }
        });

        future.join();
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateThingShadow를 참조하세요.

AWS IoT FleetWise SDK for Java 2.x를 사용한 예제

다음 코드 예제에서는를와 AWS SDK for Java 2.x 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS IoT FleetWise.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시작하기

안녕하세요 AWS IoT FleetWise

다음 코드 예제에서는 AWS IoT FleetWise를 사용하여 시작하는 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
public class HelloFleetwise {

        public static void main(String[] args) {
            ListSignalCatalogs();
```

```
        }

        public static void  ListSignalCatalogs() {
            try (IoTFleetWiseClient fleetWiseClient = IoTFleetWiseClient.builder()
                    .region(Region.US_EAST_1)
                    .credentialsProvider(DefaultCredentialsProvider.create())
                    .build()) {

                ListSignalCatalogsRequest request =
ListSignalCatalogsRequest.builder()
                        .maxResults(10) // Optional: limit per page
                        .build();

                ListSignalCatalogsIterable paginator =
fleetWiseClient.listSignalCatalogsPaginator(request);
                boolean found = false;

                for (ListSignalCatalogsResponse response : paginator) {
                    for (SignalCatalogSummary summary : response.summaries()) {
                        found = true;
                        System.out.println("Catalog Name: " + summary.name());
                        System.out.println("ARN: " + summary.arn());
                        System.out.println("Created: " + summary.creationTime());
                        System.out.println("Last Modified: " +
summary.lastModificationTime());
                        System.out.println("---------------");
                    }
                }

                if (!found) {
                    System.out.println("No AWS Fleetwise Signal Catalogs were
found.");
                }

            } catch (IoTFleetWiseException e) {
                System.err.println("Error listing signal catalogs: " +
e.awsErrorDetails().errorMessage());
                throw new RuntimeException(e);
            }
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 listSignalCatalogsPaginator를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 표준화된 신호 모음을 만듭니다.
- 차량 그룹을 나타내는 플릿을 만듭니다.
- 매니페스트를 새로 만듭니다.
- 디코더 매니페스트를 생성합니다.
- 모델 매니페스트의 상태를 확인합니다.
- 디코더의 상태를 확인합니다.
- IoT 사물을 만듭니다.
- 차량을 만듭니다.
- 차량 세부 정보를 표시합니다.
- AWS IoT FleetWise 자산을 삭제합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

AWS IoT SiteWise 기능을 보여주는 대화형 시나리오를 실행합니다.

```
public class FleetwiseScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    static FleetwiseActions actions = new FleetwiseActions();
    private static final Logger logger =
 LoggerFactory.getLogger(FleetwiseScenario.class);
    static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        final String usage =
                """
                Usage:
```

```
                    <signalCatalogName> <manifestName> <fleetId> <vecName> <decName>

                Where:
                    signalCatalogName     - The name of the Signal Catalog to create
(eg, catalog30).
                    manifestName          - The name of the Vehicle Model (Model
Manifest) to create (eg, manifest30).
                    fleetId               - The ID of the Fleet to create (eg,
fleet30).
                    vecName               - The name of the Vehicle to create (eg,
vehicle30).
                    decName               - The name of the Decoder Manifest to
create (eg, decManifest30).
                """;

        if (args.length != 5) {
            logger.info(usage);
            return;
        }

        String signalCatalogName = args[0];
        String manifestName = args[1];
        String fleetId = args[2];
        String vecName = args[3];
        String decName = args[4];

        logger.info(
                """
                 AWS IoT FleetWise is a managed service that simplifies the
                 process of collecting, organizing, and transmitting vehicle
                 data to the cloud in near real-time. Designed for automakers
                 and fleet operators, it allows you to define vehicle models,
                 specify the exact data you want to collect (such as engine
                 temperature, speed, or battery status), and send this data to
                 AWS for analysis. By using intelligent data collection
                 techniques, IoT FleetWise reduces the volume of data
                 transmitted by filtering and transforming it at the edge,
                 helping to minimize bandwidth usage and costs.

                At its core, AWS IoT FleetWise helps organizations build
                scalable systems for vehicle data management and analytics,
                supporting a wide variety of vehicles and sensor configurations.
                You can define signal catalogs and decoder manifests that describe
                how raw CAN bus signals are translated into readable data, making
```

```
                the platform highly flexible and extensible. This allows
                manufacturers to optimize vehicle performance, improve safety,
                and reduce maintenance costs by gaining real-time visibility
                into fleet operations.
                """);

        waitForInputToContinue(scanner);
        logger.info(DASHES);
        try {
            runScenario(signalCatalogName, manifestName, fleetId, vecName, decName);
        } catch (RuntimeException e) {
            logger.info(e.getMessage());
        }
    }

    private static void runScenario(String signalCatalogName,
                                    String manifestName,
                                    String fleetId,
                                    String vecName,
                                    String decName) {
        logger.info(DASHES);
        logger.info("1. Creates a collection of standardized signals that can be
reused to create vehicle models");
        String signalCatalogArn;
        try {
            signalCatalogArn =
actions.createSignalCatalogAsync(signalCatalogName).join();
            logger.info("The collection ARN is " + signalCatalogArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ValidationException) {
                logger.error("The request failed due to a validation issue: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Create a fleet that represents a group of vehicles");
        logger.info(
```

```
                """
                Creating an IoT FleetWise fleet allows you to efficiently collect,
                organize, and transfer vehicle data to the cloud, enabling real-
time
                insights into vehicle performance and health.

                It helps reduce data costs by allowing you to filter and prioritize
                only the most relevant vehicle signals, supporting advanced
 analytics
                and predictive maintenance use cases.
                """);

        waitForInputToContinue(scanner);
        String fleetid;
        try {
            fleetid = actions.createFleetAsync(signalCatalogArn, fleetId).join();
            logger.info("The fleet Id is " + fleetid);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The resource was not found: {}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Create a model manifest");
        logger.info(
                """
                An AWS IoT FleetWise manifest defines the structure and
                relationships of vehicle data. The model manifest specifies
                which signals to collect and how they relate to vehicle systems,
                while the decoder manifest defines how to decode raw vehicle data
                into meaningful signals.
                """);
        waitForInputToContinue(scanner);
        String manifestArn;
        try {
            List<Node> nodes =
 actions.listSignalCatalogNodeAsync(signalCatalogName).join();
```

```
            manifestArn = actions.createModelManifestAsync(manifestName,
signalCatalogArn, nodes).join();
            logger.info("The manifest ARN is {}", manifestArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            logger.error("An unexpected error occurred.", cause);
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Create a decoder manifest");
        logger.info(
                """
                A decoder manifest in AWS IoT FleetWise defines how raw vehicle
                data (such as CAN signals) should be interpreted and decoded
                into meaningful signals. It acts as a translation layer
                that maps vehicle-specific protocols to standardized data formats
                using decoding rules. This is crucial for extracting usable
                data from different vehicle models, even when their data
                formats vary.

                """);
        waitForInputToContinue(scanner);
        String decArn;
        try {
            decArn = actions.createDecoderManifestAsync(decName,
manifestArn).join();
            logger.info("The decoder manifest ARN is {}", decArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            logger.error("An unexpected error occurred.", cause);
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("5. Check the status of the model manifest");
        logger.info(
                """
                The model manifest must be in an ACTIVE state before it can be used
                to create or update a vehicle.
                """);
```

```
        waitForInputToContinue(scanner);
        try {
            actions.updateModelManifestAsync(manifestName);
            actions.waitForModelManifestActiveAsync(manifestName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            logger.error("An unexpected error occurred while waiting for the model
manifest status.", cause);
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("6. Check the status of the decoder");
        logger.info(
                """
                The decoder manifest must be in an ACTIVE state before it can be
used
                to create or update a vehicle.
                """);
        waitForInputToContinue(scanner);
        try {
            actions.updateDecoderManifestAsync(decName);
            actions.waitForDecoderManifestActiveAsync(decName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            logger.error("An unexpected error occurred while waiting for the decoder
manifest status.", cause);
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Create an IoT Thing");
        logger.info(
                """
                AWS IoT FleetWise expects an existing AWS IoT Thing with the same
                name as the vehicle name you are passing to createVehicle method.
                Before calling createVehicle(), you must create an AWS IoT Thing
                with the same name using the AWS IoT Core service.
                """);
        waitForInputToContinue(scanner);
        try {
```

```
            actions.createThingIfNotExistsAsync(vecName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceAlreadyExistsException) {
                logger.error("The resource exists: {}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Create a vehicle");
        logger.info(
                """
                Creating a vehicle in AWS IoT FleetWise allows you to digitally
                represent and manage a physical vehicle within the AWS ecosystem.
                This enables efficient ingestion, transformation, and transmission
                of vehicle telemetry data to the cloud for analysis.
                """);
        waitForInputToContinue(scanner);
        try {
            actions.createVehicleAsync(vecName, manifestArn, decArn).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();

            if (cause instanceof ResourceNotFoundException) {
                logger.error("The required resource was not found: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred while creating vehicle.",
cause);
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("9. Display vehicle details");
        waitForInputToContinue(scanner);
        try {
            actions.getVehicleDetailsAsync(vecName).join();
```

```
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The resource was not found: {}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("10. Delete the AWS IoT Fleetwise Assets");
        logger.info("Would you like to delete the IoT Fleetwise Assets? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            try {
                actions.deleteVehicleAsync(vecName).join();
                actions.deleteDecoderManifestAsync(decName).join();
                actions.deleteModelManifestAsync(manifestName).join();
                actions.deleteFleetAsync(fleetid).join();
                actions.deleteSignalCatalogAsync(signalCatalogName).join();
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    // Handle the case where the resource is not found.
                    logger.error("The resource was not found: {}",
cause.getMessage());
                } else if (cause instanceof RuntimeException) {
                    // Handle other runtime exceptions.
                    logger.error("An unexpected error occurred: {}",
cause.getMessage());
                } else {
                    // Catch any other unexpected exceptions.
                    logger.error("An unknown error occurred.", cause);
                }
                return;
            }

            logger.info(DASHES);
            logger.info(
                    """
```

```
                    Thank you for checking out the AWS IoT Fleetwise Service Use
 demo. We hope you
                    learned something new, or got some inspiration for your own apps
 today.
                    For more AWS code examples, have a look at:
                    https://docs.aws.amazon.com/code-library/latest/ug/what-is-code-
library.html
                    """);
            logger.info(DASHES);
        } else {
            logger.info("The AWS resources will not be deleted.");
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

AWS IoT FleetWise SDK 메서드의 래퍼 클래스입니다.

```
public class FleetwiseActions {
    private static final Logger logger =
 LoggerFactory.getLogger(FleetwiseActions.class);
    private static IoTFleetWiseAsyncClient ioTFleetWiseAsyncClient;

    private static IoTFleetWiseAsyncClient getAsyncClient() {
        if (ioTFleetWiseAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                    .maxConcurrency(100)
```

```
                    .connectionTimeout(Duration.ofSeconds(60))
                    .readTimeout(Duration.ofSeconds(60))
                    .writeTimeout(Duration.ofSeconds(60))
                    .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
                    .retryStrategy(RetryMode.STANDARD)
                    .build();

            ioTFleetWiseAsyncClient = IoTFleetWiseAsyncClient.builder()
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
        return ioTFleetWiseAsyncClient;
    }


    /**
     * Creates a signal catalog.
     *
     * @param signalCatalogName the name of the signal catalog to be created
     * @return a {@link CompletableFuture} that completes with the Amazon Resource
Name (ARN) of the created signal catalog
     */
    public CompletableFuture<String> createSignalCatalogAsync(String
signalCatalogName) {
        return deleteSignalCatalogIfExistsAsync(signalCatalogName)
                .thenCompose(ignored -> delayAsync(2000)) // Wait for 2 seconds
                .thenCompose(ignored -> {
                    List<Node> nodes = List.of(
                            Node.builder().branch(
                                    Branch.builder()
                                            .fullyQualifiedName("Vehicle")
                                            .description("Root branch")
                                            .build()
                            ).build(),
                            Node.builder().branch(
                                    Branch.builder()

.fullyQualifiedName("Vehicle.Powertrain")
```

```
                                        .description("Powertrain branch")
                                        .build()
                            ).build(),
                            Node.builder().sensor(
                                    Sensor.builder()

.fullyQualifiedName("Vehicle.Powertrain.EngineRPM")
                                        .description("Engine RPM")
                                        .dataType(NodeDataType.DOUBLE)
                                        .unit("rpm")
                                        .build()
                            ).build(),
                            Node.builder().sensor(
                                    Sensor.builder()

.fullyQualifiedName("Vehicle.Powertrain.VehicleSpeed")
                                        .description("Vehicle Speed")
                                        .dataType(NodeDataType.DOUBLE)
                                        .unit("km/h")
                                        .build()
                            ).build()
                    );

                    CreateSignalCatalogRequest request =
CreateSignalCatalogRequest.builder()
                            .name(signalCatalogName)
                            .nodes(nodes)
                            .build();

                    CompletableFuture<String> result = new CompletableFuture<>();

                    getAsyncClient().createSignalCatalog(request)
                            .whenComplete((response, exception) -> {
                                if (exception != null) {
                                    Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                                    if (cause instanceof ValidationException) {
                                        result.completeExceptionally(cause);
                                    } else {
                                        result.completeExceptionally(new
RuntimeException("Error creating the catalog", cause));
                                    }
                                } else {
```

```
                                    result.complete(response.arn());
                                }
                            });

                    return result;
                });
    }

    /**
     * Delays the execution of the current thread asynchronously for the specified
duration.
     *
     * @param millis the duration of the delay in milliseconds
     * @return a {@link CompletableFuture} that completes after the specified delay
     */
    private static CompletableFuture<Void> delayAsync(long millis) {
        return CompletableFuture.runAsync(() -> {
            try {
                Thread.sleep(millis);
            } catch (InterruptedException e) {
                throw new CompletionException("Sleep interrupted", e);
            }
        });
    }

    /**
     * Deletes the specified signal catalog.
     *
     * @param signalCatalogName the name of the signal catalog to delete
     * @return a {@link CompletableFuture} representing the asynchronous operation.
     */
    public static CompletableFuture<Void> deleteSignalCatalogIfExistsAsync(String
signalCatalogName) {
        DeleteSignalCatalogRequest request = DeleteSignalCatalogRequest.builder()
                .name(signalCatalogName)
                .build();

        return getAsyncClient().deleteSignalCatalog(request)
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ResourceNotFoundException) {
```

```
                            throw new CompletionException(new
RuntimeException("Signal Catalog not found: " + signalCatalogName));
                        }
                        throw new RuntimeException("Failed to delete signal catalog:
" + signalCatalogName, cause);
                    }
                    return null;
                });
    }


    /**
     * Creates a new decoder manifest.
     *
     * @param name             the name of the decoder manifest
     * @param modelManifestArn the ARN of the model manifest
     * @return a {@link CompletableFuture} that completes with the ARN of the
created decoder manifest
     */
    public CompletableFuture<String> createDecoderManifestAsync(String name, String
modelManifestArn) {
        String interfaceId = "can0";
        NetworkInterface networkInterface = NetworkInterface.builder()
                .interfaceId(interfaceId)
                .type(NetworkInterfaceType.CAN_INTERFACE)
                .canInterface(CanInterface.builder()
                        .name("canInterface0")
                        .protocolName("CAN")
                        .protocolVersion("1.0")
                        .build())
                .build();

        // Vehicle.Powertrain.EngineRPM decoder.
        SignalDecoder engineRpmDecoder = SignalDecoder.builder()
                .fullyQualifiedName("Vehicle.Powertrain.EngineRPM")
                .interfaceId(interfaceId)
                .type(SignalDecoderType.CAN_SIGNAL)
                .canSignal(CanSignal.builder()
                        .messageId(100)
                        .isBigEndian(false)
                        .isSigned(false)
                        .startBit(0)
                        .length(16)
                        .factor(1.0)
```

```
                        .offset(0.0)
                        .build())
                .build();

        // Vehicle.Powertrain.VehicleSpeed decoder.
        SignalDecoder vehicleSpeedDecoder = SignalDecoder.builder()
                .fullyQualifiedName("Vehicle.Powertrain.VehicleSpeed")
                .interfaceId(interfaceId)
                .type(SignalDecoderType.CAN_SIGNAL)
                .canSignal(CanSignal.builder()
                        .messageId(101)
                        .isBigEndian(false)
                        .isSigned(false)
                        .startBit(16)
                        .length(16)
                        .factor(1.0)
                        .offset(0.0)
                        .build())
                .build();

        CreateDecoderManifestRequest request =
CreateDecoderManifestRequest.builder()
                .name(name)
                .modelManifestArn(modelManifestArn)
                .networkInterfaces(List.of(networkInterface))
                .signalDecoders(List.of(engineRpmDecoder, vehicleSpeedDecoder))
                .build();

        CompletableFuture<String> result = new CompletableFuture<>();

        getAsyncClient().createDecoderManifest(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof DecoderManifestValidationException) {
                            result.completeExceptionally(new
CompletionException("The request contains signal decoders with validation errors: "
+ cause.getMessage(), cause));
                        } else {
                            result.completeExceptionally(new
CompletionException("Failed to create decoder manifest: " + exception.getMessage(),
exception));
```

```
                        }
                    } else {
                        result.complete(response.arn()); // Complete successfully
with the ARN
                    }
                });

        return result;
    }

    /**
     * Deletes a decoder manifest.
     *
     * @param name the name of the decoder manifest to delete
     * @return a {@link CompletableFuture} that completes when the decoder manifest
has been deleted
     */
    public CompletableFuture<Void> deleteDecoderManifestAsync(String name) {
        return
getAsyncClient().deleteDecoderManifest(DeleteDecoderManifestRequest.builder().name(name).bu
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }
                        throw new RuntimeException("Failed to delete the decoder
manifest: " + cause);
                    }
                    return null;
                });
    }


    /**
     * Deletes a vehicle with the specified name.
     *
     * @param vecName the name of the vehicle to be deleted
     * @return a {@link CompletableFuture} that completes when the vehicle has been
deleted
     */
    public CompletableFuture<Void> deleteVehicleAsync(String vecName) {
        DeleteVehicleRequest request = DeleteVehicleRequest.builder()
```

```
                .vehicleName(vecName)
                .build();

        return getAsyncClient().deleteVehicle(request)
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }
                        throw new RuntimeException("Failed to delete the vehicle: "
+ cause);
                    }
                    return null;
                });
    }


    /**
     * Updates the model manifest.
     *
     * @param name the name of the model manifest to update
     */
    public void updateModelManifestAsync(String name) {
        UpdateModelManifestRequest request = UpdateModelManifestRequest.builder()
                .name(name)
                .status(ManifestStatus.ACTIVE)
                .build();

        getAsyncClient().updateModelManifest(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        throw new CompletionException("Failed to update model
manifest: " + exception.getMessage(), exception);
                    }
                })
                .thenApply(response -> null);
    }


    /**
     * Updates the decoder manifest with the given name.
     *
```

```
    * @param name the name of the decoder manifest to update
    * @return a {@link CompletableFuture} that completes when the update operation
is finished
    */
   public CompletableFuture<Void> updateDecoderManifestAsync(String name) {
       UpdateDecoderManifestRequest request =
UpdateDecoderManifestRequest.builder()
               .name(name)
               .status(ManifestStatus.ACTIVE)
               .build();

       return getAsyncClient().updateDecoderManifest(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       throw new CompletionException("Failed to update decoder
manifest: " + exception.getMessage(), exception);
                   }
               })
               .thenApply(response -> null);
   }


   /**
    * Creates a new vehicle in the system.
    *
    * @param vecName     the name of the vehicle to be created
    * @param manifestArn the Amazon Resource Name (ARN) of the model manifest for
the vehicle
    * @param decArn      the Amazon Resource Name (ARN) of the decoder manifest for
the vehicle
    * @return a {@link CompletableFuture} that completes when the vehicle has been
created, or throws a
    */
   public CompletableFuture<Void> createVehicleAsync(String vecName, String
manifestArn, String decArn) {
       CreateVehicleRequest request = CreateVehicleRequest.builder()
               .vehicleName(vecName)
               .modelManifestArn(manifestArn)
               .decoderManifestArn(decArn)
               .build();

       CompletableFuture<Void> result = new CompletableFuture<>();
       getAsyncClient().createVehicle(request)
               .whenComplete((response, exception) -> {
```

```
                    if (exception != null) {
                        Throwable cause = exception instanceof CompletionException ?
exception.getCause() : exception;

                        if (cause instanceof ResourceNotFoundException) {
                            result.completeExceptionally(cause);
                        } else {
                            result.completeExceptionally(new
RuntimeException("Failed to create vehicle: " + cause.getMessage(), cause));
                        }
                    } else {
                        logger.info("Vehicle '{}' created successfully.", vecName);
                        result.complete(null); // mark future as complete
                    }
                });

        return result;
    }


    /**
     * Waits for the decoder manifest to become active.
     *
     * @param decoderName the name of the decoder to wait for
     * @return a {@link CompletableFuture} that completes when the decoder manifest
becomes active, or exceptionally if an error occurs or the manifest becomes invalid
     */
    public CompletableFuture<Void> waitForDecoderManifestActiveAsync(String
decoderName) {
        CompletableFuture<Void> result = new CompletableFuture<>();

        ScheduledExecutorService scheduler =
Executors.newSingleThreadScheduledExecutor();
        AtomicInteger secondsElapsed = new AtomicInteger(0);
        AtomicReference<ManifestStatus> lastStatus = new
AtomicReference<>(ManifestStatus.DRAFT);

        logger.info(" Elapsed: 0s | Decoder Status: DRAFT");

        final Runnable pollTask = new Runnable() {
            @Override
            public void run() {
                int elapsed = secondsElapsed.incrementAndGet();
```

```
                // Check status every 5 seconds
                if (elapsed % 5 == 0) {
                    GetDecoderManifestRequest request =
GetDecoderManifestRequest.builder()
                            .name(decoderName)
                            .build();

                    getAsyncClient().getDecoderManifest(request)
                            .whenComplete((response, exception) -> {
                                if (exception != null) {
                                    Throwable cause = exception instanceof
CompletionException ? exception.getCause() : exception;

                                    scheduler.shutdown();
                                    if (cause instanceof ResourceNotFoundException)
{
                                        result.completeExceptionally(new
RuntimeException("Decoder manifest not found: " + cause.getMessage(), cause));
                                    } else {
                                        result.completeExceptionally(new
RuntimeException("Error while polling decoder manifest status: " +
exception.getMessage(), exception));
                                    }
                                    return;
                                }

                                ManifestStatus status = response.status();
                                lastStatus.set(status);

                                if (status == ManifestStatus.ACTIVE) {
                                    logger.info("\r Elapsed: {}s | Decoder Status:
ACTIVE", elapsed);
                                    scheduler.shutdown();
                                    result.complete(null);
                                } else if (status == ManifestStatus.INVALID) {
                                    logger.info("\r Elapsed: {}s | Decoder Status:
INVALID", elapsed);
                                    scheduler.shutdown();
                                    result.completeExceptionally(new
RuntimeException("Decoder manifest became INVALID. Cannot proceed."));
                                } else {
                                    logger.info("\r# Elapsed: {}s | Decoder Status:
{}", elapsed, status);
                                }
```

```
                            });
                } else {
                    logger.info("\r Elapsed: {}s | Decoder Status: {}", elapsed,
lastStatus.get());
                }
            }
        };

        // Start the task with an initial delay of 1 second, and repeat every second
        scheduler.scheduleAtFixedRate(pollTask, 1, 1, TimeUnit.SECONDS);
        return result;
    }



    /**
     * Waits for the specified model manifest to become active.
     *
     * @param manifestName the name of the model manifest to wait for
     */
    public CompletableFuture<Void> waitForModelManifestActiveAsync(String
manifestName) {
        CompletableFuture<Void> result = new CompletableFuture<>();

        ScheduledExecutorService scheduler =
Executors.newSingleThreadScheduledExecutor();
        AtomicInteger secondsElapsed = new AtomicInteger(0);
        AtomicReference<ManifestStatus> lastStatus = new
AtomicReference<>(ManifestStatus.DRAFT);

        logger.info("Elapsed: 0s | Status: DRAFT");

        final Runnable pollTask = new Runnable() {
            @Override
            public void run() {
                int elapsed = secondsElapsed.incrementAndGet();

                // Only check status every 5 seconds
                if (elapsed % 5 == 0) {
                    GetModelManifestRequest request =
GetModelManifestRequest.builder()
                            .name(manifestName)
                            .build();
```

```
                    getAsyncClient().getModelManifest(request)
                            .whenComplete((response, exception) -> {
                                if (exception != null) {
                                    Throwable cause = exception instanceof
CompletionException ? exception.getCause() : exception;

                                    scheduler.shutdown();
                                    if (cause instanceof ResourceNotFoundException)
{
                                        result.completeExceptionally(new
RuntimeException("Model manifest not found: " + cause.getMessage(), cause));
                                    } else {
                                        result.completeExceptionally(new
RuntimeException("Error while polling model manifest status: " +
exception.getMessage(), exception));
                                    }
                                    return;
                                }

                                ManifestStatus status = response.status();
                                lastStatus.set(status);

                                if (status == ManifestStatus.ACTIVE) {
                                    logger.info("\rElapsed: {}s | Status: ACTIVE",
elapsed);
                                    scheduler.shutdown();
                                    result.complete(null);
                                } else if (status == ManifestStatus.INVALID) {
                                    logger.info("\rElapsed: {}s | Status: INVALID",
elapsed);
                                    scheduler.shutdown();
                                    result.completeExceptionally(new
RuntimeException("Model manifest became INVALID. Cannot proceed."));
                                } else {
                                    logger.info("\rElapsed: {}s | Status: {}",
elapsed, status);
                                }
                            });
                } else {
                    logger.info("\rElapsed: {}s | Status: {}", elapsed,
lastStatus.get());
                }
            }
        };
```

```
        // Start the task with an initial delay of 1 second, and repeat every second
        scheduler.scheduleAtFixedRate(pollTask, 1, 1, TimeUnit.SECONDS);
        return result;
    }



    /**
     * Fetches the details of a vehicle.
     *
     * @param vehicleName the name of the vehicle to fetch details for
     * @return a {@link CompletableFuture} that completes when the vehicle details
have been fetched
     */
    public CompletableFuture<Void> getVehicleDetailsAsync(String vehicleName) {
        GetVehicleRequest request = GetVehicleRequest.builder()
                .vehicleName(vehicleName)
                .build();

        CompletableFuture<Void> result = new CompletableFuture<>();

        getAsyncClient().getVehicle(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception instanceof CompletionException ?
exception.getCause() : exception;

                        if (cause instanceof ResourceNotFoundException) {
                            result.completeExceptionally(cause); // don't rewrap
                        } else {
                            result.completeExceptionally(new
RuntimeException("Failed to fetch vehicle details: " + cause.getMessage(), cause));
                        }
                    } else {
                        Map<String, Object> details = new HashMap<>();
                        details.put("vehicleName", response.vehicleName());
                        details.put("arn", response.arn());
                        details.put("modelManifestArn",
response.modelManifestArn());
                        details.put("decoderManifestArn",
response.decoderManifestArn());
                        details.put("attributes", response.attributes());
```

```
                        details.put("creationTime",
response.creationTime().toString());
                        details.put("lastModificationTime",
response.lastModificationTime().toString());

                        logger.info("Vehicle Details:");
                        details.forEach((key, value) -> logger.info("• {} : {}",
key, value));

                        result.complete(null); // mark as successful
                    }
                });

        return result;
    }


    /**
     * Creates an IoT Thing if it does not already exist.
     *
     * @param thingName the name of the IoT Thing to create
     * @return a {@link CompletableFuture} that completes when the IoT Thing has
been created or if it already exists
     */
    public CompletableFuture<Void> createThingIfNotExistsAsync(String thingName) {
        IotAsyncClient iotClient = IotAsyncClient.builder()
                .region(Region.US_EAST_1)
                .build();

        CreateThingRequest request = CreateThingRequest.builder()
                .thingName(thingName)
                .build();

        return iotClient.createThing(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        if (exception instanceof ResourceAlreadyExistsException) {
                            logger.info(" IoT Thing already exists: " + thingName);
                        } else {
                            throw new CompletionException("Failed to create IoT
Thing: " + thingName, exception);
                        }
                    } else {
                        logger.info("IoT Thing created: " + response.thingName());
```

```
                    }
                })
                .thenApply(response -> null);
    }


    /**
     * Deletes a model manifest.
     *
     * @param name the name of the model manifest to delete
     * @return a {@link CompletableFuture} that completes when the model manifest
has been deleted
     */
    public CompletableFuture<Void> deleteModelManifestAsync(String name) {
        DeleteModelManifestRequest request = DeleteModelManifestRequest.builder()
                .name(name)
                .build();

        return getAsyncClient().deleteModelManifest(request)
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }
                        throw new RuntimeException("Failed to delete the model
manifest: " + cause);
                    }
                    logger.info("{} was successfully deleted", name);
                    return null;
                });
    }


    /**
     * Deletes a signal catalog.
     *
     * @param name the name of the signal catalog to delete
     * @return a {@link CompletableFuture} that completes when the signal catalog is
deleted
     */
    public CompletableFuture<Void> deleteSignalCatalogAsync(String name) {
        DeleteSignalCatalogRequest request = DeleteSignalCatalogRequest.builder()
```

```
                .name(name)
                .build();

        return getAsyncClient().deleteSignalCatalog(request)
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }
                        throw new RuntimeException("Failed to delete the signal
catalog: " + cause);
                    }
                    logger.info("{} was successfully deleted", name);
                    return null;
                });
    }

    /**
     * Asynchronously retrieves a list of all nodes in the specified signal catalog.
     *
     * @param signalCatalogName the name of the signal catalog to retrieve nodes for
     * @return a {@link CompletableFuture} that, when completed, contains a {@link
List} of {@link Node} objects
     * representing all the nodes in the specified signal catalog
     */
    public CompletableFuture<List<Node>> listSignalCatalogNodeAsync(String
signalCatalogName) {
        ListSignalCatalogNodesRequest request =
ListSignalCatalogNodesRequest.builder()
                .name(signalCatalogName)
                .build();

        List<Node> allNodes = new ArrayList<>();

        return getAsyncClient().listSignalCatalogNodesPaginator(request)
                .subscribe(response -> allNodes.addAll(response.nodes()))
                .thenApply(v -> allNodes);
    }



    /**
```

```
    * Creates a model manifest.
    *
    * @param name             the name of the model manifest to create
    * @param signalCatalogArn the Amazon Resource Name (ARN) of the signal catalog
    * @param nodes            a list of nodes to include in the model manifest
    * @return a {@link CompletableFuture} that completes with the ARN of the
created model manifest
    */
   public CompletableFuture<String> createModelManifestAsync(String name,
                                                              String
signalCatalogArn,
                                                              List<Node> nodes) {
       // Extract the fully qualified names (FQNs) from each Node in the provided
list.
       List<String> fqnList = nodes.stream()
               .map(node -> {
                   if (node.sensor() != null) {
                       return node.sensor().fullyQualifiedName();
                   } else if (node.branch() != null) {
                       return node.branch().fullyQualifiedName();
                   } else if (node.attribute() != null) {
                       return node.attribute().fullyQualifiedName();
                   } else {
                       throw new RuntimeException("Unsupported node type");
                   }
               })
               .toList();

       CreateModelManifestRequest request = CreateModelManifestRequest.builder()
               .name(name)
               .signalCatalogArn(signalCatalogArn)
               .nodes(fqnList)
               .build();


       CompletableFuture<String> result = new CompletableFuture<>();
       getAsyncClient().createModelManifest(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                       if (cause instanceof InvalidSignalsException) {
```

```
                            result.completeExceptionally(new
CompletionException("The request contains signals that aren't valid: " +
cause.getMessage(), cause));
                        } else {
                            result.completeExceptionally(new
CompletionException("Failed to create model manifest: " + exception.getMessage(),
exception));
                        }
                    } else {
                        result.complete(response.arn()); // Complete successfully
with the ARN
                    }
                });

        return result;
    }


    /**
     * Deletes a fleet based on the provided fleet ID.
     *
     * @param fleetId the ID of the fleet to be deleted
     */
    public CompletableFuture<Void> deleteFleetAsync(String fleetId) {
        DeleteFleetRequest request = DeleteFleetRequest.builder()
                .fleetId(fleetId)
                .build();

        return getAsyncClient().deleteFleet(request)
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }
                        throw new RuntimeException("Failed to delete the fleet: " +
cause);
                    }
                    logger.info("{} was successfully deleted", fleetId);
                    return null;
                });
    }
```

```
    /**
     * Creates a new fleet.
     *
     * @param catARN  the Amazon Resource Name (ARN) of the signal catalog to
 associate with the fleet
     * @param fleetId the unique identifier for the fleet
     * @return a {@link CompletableFuture} that completes with the ID of the created
 fleet
     */
    public CompletableFuture<String> createFleetAsync(String catARN, String fleetId)
 {
        CreateFleetRequest fleetRequest = CreateFleetRequest.builder()
                .fleetId(fleetId)
                .signalCatalogArn(catARN)
                .description("Built using the AWS For Java V2")
                .build();

        CompletableFuture<String> result = new CompletableFuture<>();
        getAsyncClient().createFleet(fleetRequest)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
 exception.getCause() : exception;

                        if (cause instanceof ResourceNotFoundException) {
                            result.completeExceptionally(cause);
                        } else {
                            result.completeExceptionally(new RuntimeException("An
 unexpected error occurred", cause));
                        }
                    } else {
                        result.complete(response.id());
                    }
                });

        return result;
    }
}
```

작업

`createDecoderManifest`

다음 코드 예시는 `createDecoderManifest`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Creates a new decoder manifest.
     *
     * @param name             the name of the decoder manifest
     * @param modelManifestArn the ARN of the model manifest
     * @return a {@link CompletableFuture} that completes with the ARN of the
created decoder manifest
     */
    public CompletableFuture<String> createDecoderManifestAsync(String name, String
modelManifestArn) {
        String interfaceId = "can0";
        NetworkInterface networkInterface = NetworkInterface.builder()
                .interfaceId(interfaceId)
                .type(NetworkInterfaceType.CAN_INTERFACE)
                .canInterface(CanInterface.builder()
                        .name("canInterface0")
                        .protocolName("CAN")
                        .protocolVersion("1.0")
                        .build())
                .build();

        // Vehicle.Powertrain.EngineRPM decoder.
        SignalDecoder engineRpmDecoder = SignalDecoder.builder()
                .fullyQualifiedName("Vehicle.Powertrain.EngineRPM")
                .interfaceId(interfaceId)
                .type(SignalDecoderType.CAN_SIGNAL)
                .canSignal(CanSignal.builder()
                        .messageId(100)
```

```
                        .isBigEndian(false)
                        .isSigned(false)
                        .startBit(0)
                        .length(16)
                        .factor(1.0)
                        .offset(0.0)
                        .build())
                .build();

        // Vehicle.Powertrain.VehicleSpeed decoder.
        SignalDecoder vehicleSpeedDecoder = SignalDecoder.builder()
                .fullyQualifiedName("Vehicle.Powertrain.VehicleSpeed")
                .interfaceId(interfaceId)
                .type(SignalDecoderType.CAN_SIGNAL)
                .canSignal(CanSignal.builder()
                        .messageId(101)
                        .isBigEndian(false)
                        .isSigned(false)
                        .startBit(16)
                        .length(16)
                        .factor(1.0)
                        .offset(0.0)
                        .build())
                .build();

        CreateDecoderManifestRequest request =
CreateDecoderManifestRequest.builder()
                .name(name)
                .modelManifestArn(modelManifestArn)
                .networkInterfaces(List.of(networkInterface))
                .signalDecoders(List.of(engineRpmDecoder, vehicleSpeedDecoder))
                .build();

        CompletableFuture<String> result = new CompletableFuture<>();

        getAsyncClient().createDecoderManifest(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof DecoderManifestValidationException) {
```

```
                            result.completeExceptionally(new
CompletionException("The request contains signal decoders with validation errors: "
+ cause.getMessage(), cause));
                        } else {
                            result.completeExceptionally(new
CompletionException("Failed to create decoder manifest: " + exception.getMessage(),
exception));
                        }
                    } else {
                        result.complete(response.arn()); // Complete successfully
with the ARN
                    }
                });

        return result;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 createDecoderManifest를 참조하세요.

`createFleet`

다음 코드 예시는 `createFleet`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Creates a new fleet.
     *
     * @param catARN  the Amazon Resource Name (ARN) of the signal catalog to
associate with the fleet
     * @param fleetId the unique identifier for the fleet
     * @return a {@link CompletableFuture} that completes with the ID of the created
fleet
     */
```

```
    public CompletableFuture<String> createFleetAsync(String catARN, String fleetId)
{
        CreateFleetRequest fleetRequest = CreateFleetRequest.builder()
                .fleetId(fleetId)
                .signalCatalogArn(catARN)
                .description("Built using the AWS For Java V2")
                .build();

        CompletableFuture<String> result = new CompletableFuture<>();
        getAsyncClient().createFleet(fleetRequest)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ResourceNotFoundException) {
                            result.completeExceptionally(cause);
                        } else {
                            result.completeExceptionally(new RuntimeException("An
unexpected error occurred", cause));
                        }
                    } else {
                        result.complete(response.id());
                    }
                });

        return result;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 createFleet을 참조하세요.

`createModelManifest`

다음 코드 예시는 `createModelManifest`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Creates a model manifest.
    *
    * @param name             the name of the model manifest to create
    * @param signalCatalogArn the Amazon Resource Name (ARN) of the signal catalog
    * @param nodes            a list of nodes to include in the model manifest
    * @return a {@link CompletableFuture} that completes with the ARN of the
created model manifest
    */
   public CompletableFuture<String> createModelManifestAsync(String name,
                                                              String
signalCatalogArn,
                                                              List<Node> nodes) {
       // Extract the fully qualified names (FQNs) from each Node in the provided
list.
       List<String> fqnList = nodes.stream()
               .map(node -> {
                   if (node.sensor() != null) {
                       return node.sensor().fullyQualifiedName();
                   } else if (node.branch() != null) {
                       return node.branch().fullyQualifiedName();
                   } else if (node.attribute() != null) {
                       return node.attribute().fullyQualifiedName();
                   } else {
                       throw new RuntimeException("Unsupported node type");
                   }
               })
               .toList();

       CreateModelManifestRequest request = CreateModelManifestRequest.builder()
               .name(name)
               .signalCatalogArn(signalCatalogArn)
               .nodes(fqnList)
               .build();


       CompletableFuture<String> result = new CompletableFuture<>();
       getAsyncClient().createModelManifest(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
```

```
                        if (cause instanceof InvalidSignalsException) {
                            result.completeExceptionally(new
CompletionException("The request contains signals that aren't valid: " +
cause.getMessage(), cause));
                        } else {
                            result.completeExceptionally(new
CompletionException("Failed to create model manifest: " + exception.getMessage(),
exception));
                        }
                    } else {
                        result.complete(response.arn()); // Complete successfully
with the ARN
                    }
                });

        return result;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 createModelManifest를 참조하세요.

`createSignalCatalog`

다음 코드 예시는 `createSignalCatalog`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Creates a signal catalog.
     *
     * @param signalCatalogName the name of the signal catalog to be created
     * @return a {@link CompletableFuture} that completes with the Amazon Resource
Name (ARN) of the created signal catalog
     */
```

```
   public CompletableFuture<String> createSignalCatalogAsync(String
signalCatalogName) {
       return deleteSignalCatalogIfExistsAsync(signalCatalogName)
               .thenCompose(ignored -> delayAsync(2000)) // Wait for 2 seconds
               .thenCompose(ignored -> {
                   List<Node> nodes = List.of(
                           Node.builder().branch(
                                   Branch.builder()
                                           .fullyQualifiedName("Vehicle")
                                           .description("Root branch")
                                           .build()
                           ).build(),
                           Node.builder().branch(
                                   Branch.builder()

.fullyQualifiedName("Vehicle.Powertrain")
                                           .description("Powertrain branch")
                                           .build()
                           ).build(),
                           Node.builder().sensor(
                                   Sensor.builder()

.fullyQualifiedName("Vehicle.Powertrain.EngineRPM")
                                           .description("Engine RPM")
                                           .dataType(NodeDataType.DOUBLE)
                                           .unit("rpm")
                                           .build()
                           ).build(),
                           Node.builder().sensor(
                                   Sensor.builder()

.fullyQualifiedName("Vehicle.Powertrain.VehicleSpeed")
                                           .description("Vehicle Speed")
                                           .dataType(NodeDataType.DOUBLE)
                                           .unit("km/h")
                                           .build()
                           ).build()
                   );

                   CreateSignalCatalogRequest request =
CreateSignalCatalogRequest.builder()
                           .name(signalCatalogName)
                           .nodes(nodes)
                           .build();
```

```
                CompletableFuture<String> result = new CompletableFuture<>();

                getAsyncClient().createSignalCatalog(request)
                        .whenComplete((response, exception) -> {
                            if (exception != null) {
                                Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                                if (cause instanceof ValidationException) {
                                    result.completeExceptionally(cause);
                                } else {
                                    result.completeExceptionally(new
RuntimeException("Error creating the catalog", cause));
                                }
                            } else {
                                result.complete(response.arn());
                            }
                        });

                return result;
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 createSignalCatalog를 참조하세요.

`createVehicle`

다음 코드 예시는 `createVehicle`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
```

```
    * Creates a new vehicle in the system.
    *
    * @param vecName     the name of the vehicle to be created
    * @param manifestArn the Amazon Resource Name (ARN) of the model manifest for
the vehicle
    * @param decArn      the Amazon Resource Name (ARN) of the decoder manifest for
the vehicle
    * @return a {@link CompletableFuture} that completes when the vehicle has been
created, or throws a
    */
   public CompletableFuture<Void> createVehicleAsync(String vecName, String
manifestArn, String decArn) {
       CreateVehicleRequest request = CreateVehicleRequest.builder()
               .vehicleName(vecName)
               .modelManifestArn(manifestArn)
               .decoderManifestArn(decArn)
               .build();

       CompletableFuture<Void> result = new CompletableFuture<>();
       getAsyncClient().createVehicle(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception instanceof CompletionException ?
exception.getCause() : exception;

                       if (cause instanceof ResourceNotFoundException) {
                           result.completeExceptionally(cause);
                       } else {
                           result.completeExceptionally(new
RuntimeException("Failed to create vehicle: " + cause.getMessage(), cause));
                       }
                   } else {
                       logger.info("Vehicle '{}' created successfully.", vecName);
                       result.complete(null); // mark future as complete
                   }
               });

       return result;
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 createVehicle을 참조하세요.

deleteDecoderManifest

다음 코드 예시는 deleteDecoderManifest의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Deletes a decoder manifest.
    *
    * @param name the name of the decoder manifest to delete
    * @return a {@link CompletableFuture} that completes when the decoder manifest
has been deleted
    */
   public CompletableFuture<Void> deleteDecoderManifestAsync(String name) {
       return
getAsyncClient().deleteDecoderManifest(DeleteDecoderManifestRequest.builder().name(name).bu
               .handle((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                       if (cause instanceof ResourceNotFoundException) {
                           throw (ResourceNotFoundException) cause;
                       }
                       throw new RuntimeException("Failed to delete the decoder
manifest: " + cause);
                   }
                   return null;
               });
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 deleteDecoderManifest를 참조하세요.

deleteFleet

다음 코드 예시는 deleteFleet의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Deletes a fleet based on the provided fleet ID.
     *
     * @param fleetId the ID of the fleet to be deleted
     */
    public CompletableFuture<Void> deleteFleetAsync(String fleetId) {
        DeleteFleetRequest request = DeleteFleetRequest.builder()
                .fleetId(fleetId)
                .build();

        return getAsyncClient().deleteFleet(request)
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }
                        throw new RuntimeException("Failed to delete the fleet: " +
cause);
                    }
                    logger.info("{} was successfully deleted", fleetId);
                    return null;
                });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 deleteFleet을 참조하세요.

`deleteModelManifest`

다음 코드 예시는 `deleteModelManifest`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Deletes a model manifest.
    *
    * @param name the name of the model manifest to delete
    * @return a {@link CompletableFuture} that completes when the model manifest
has been deleted
    */
   public CompletableFuture<Void> deleteModelManifestAsync(String name) {
       DeleteModelManifestRequest request = DeleteModelManifestRequest.builder()
               .name(name)
               .build();

       return getAsyncClient().deleteModelManifest(request)
               .handle((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                       if (cause instanceof ResourceNotFoundException) {
                           throw (ResourceNotFoundException) cause;
                       }
                       throw new RuntimeException("Failed to delete the model
manifest: " + cause);
                   }
                   logger.info("{} was successfully deleted", name);
                   return null;
               });
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 deleteModelManifest를 참조하세요.

deleteSignalCatalog

다음 코드 예시는 deleteSignalCatalog의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Deletes a signal catalog.
    *
    * @param name the name of the signal catalog to delete
    * @return a {@link CompletableFuture} that completes when the signal catalog is
deleted
    */
   public CompletableFuture<Void> deleteSignalCatalogAsync(String name) {
       DeleteSignalCatalogRequest request = DeleteSignalCatalogRequest.builder()
               .name(name)
               .build();

       return getAsyncClient().deleteSignalCatalog(request)
               .handle((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                       if (cause instanceof ResourceNotFoundException) {
                           throw (ResourceNotFoundException) cause;
                       }
                       throw new RuntimeException("Failed to delete the signal
catalog: " + cause);
                   }
                   logger.info("{} was successfully deleted", name);
                   return null;
               });
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 deleteSignalCatalog를 참조하세요.

`deleteVehicle`

다음 코드 예시는 `deleteVehicle`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Deletes a vehicle with the specified name.
    *
    * @param vecName the name of the vehicle to be deleted
    * @return a {@link CompletableFuture} that completes when the vehicle has been
deleted
    */
   public CompletableFuture<Void> deleteVehicleAsync(String vecName) {
       DeleteVehicleRequest request = DeleteVehicleRequest.builder()
               .vehicleName(vecName)
               .build();

       return getAsyncClient().deleteVehicle(request)
               .handle((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                       if (cause instanceof ResourceNotFoundException) {
                           throw (ResourceNotFoundException) cause;
                       }
                       throw new RuntimeException("Failed to delete the vehicle: "
+ cause);
                   }
                   return null;
               });
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 deleteVehicle을 참조하세요.

`getDecoderManifest`

다음 코드 예시는 `getDecoderManifest`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Waits for the decoder manifest to become active.
    *
    * @param decoderName the name of the decoder to wait for
    * @return a {@link CompletableFuture} that completes when the decoder manifest
becomes active, or exceptionally if an error occurs or the manifest becomes invalid
    */
   public CompletableFuture<Void> waitForDecoderManifestActiveAsync(String
decoderName) {
       CompletableFuture<Void> result = new CompletableFuture<>();

       ScheduledExecutorService scheduler =
Executors.newSingleThreadScheduledExecutor();
       AtomicInteger secondsElapsed = new AtomicInteger(0);
       AtomicReference<ManifestStatus> lastStatus = new
AtomicReference<>(ManifestStatus.DRAFT);

       logger.info(" Elapsed: 0s | Decoder Status: DRAFT");

       final Runnable pollTask = new Runnable() {
           @Override
           public void run() {
               int elapsed = secondsElapsed.incrementAndGet();

               // Check status every 5 seconds
               if (elapsed % 5 == 0) {
```

```
                    GetDecoderManifestRequest request =
GetDecoderManifestRequest.builder()
                            .name(decoderName)
                            .build();

                    getAsyncClient().getDecoderManifest(request)
                            .whenComplete((response, exception) -> {
                                if (exception != null) {
                                    Throwable cause = exception instanceof
CompletionException ? exception.getCause() : exception;

                                    scheduler.shutdown();
                                    if (cause instanceof ResourceNotFoundException)
{
                                        result.completeExceptionally(new
RuntimeException("Decoder manifest not found: " + cause.getMessage(), cause));
                                    } else {
                                        result.completeExceptionally(new
RuntimeException("Error while polling decoder manifest status: " +
exception.getMessage(), exception));
                                    }
                                    return;
                                }

                                ManifestStatus status = response.status();
                                lastStatus.set(status);

                                if (status == ManifestStatus.ACTIVE) {
                                    logger.info("\r Elapsed: {}s | Decoder Status:
ACTIVE", elapsed);
                                    scheduler.shutdown();
                                    result.complete(null);
                                } else if (status == ManifestStatus.INVALID) {
                                    logger.info("\r Elapsed: {}s | Decoder Status:
INVALID", elapsed);
                                    scheduler.shutdown();
                                    result.completeExceptionally(new
RuntimeException("Decoder manifest became INVALID. Cannot proceed."));
                                } else {
                                    logger.info("\r# Elapsed: {}s | Decoder Status:
{}", elapsed, status);
                                }
                            });
                } else {
```

```
                    logger.info("\r Elapsed: {}s | Decoder Status: {}", elapsed,
lastStatus.get());
                }
            }
        };

        // Start the task with an initial delay of 1 second, and repeat every second
        scheduler.scheduleAtFixedRate(pollTask, 1, 1, TimeUnit.SECONDS);
        return result;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 getDecoderManifest를 참조하세요.

`getModelManifest`

다음 코드 예시는 `getModelManifest`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Waits for the specified model manifest to become active.
     *
     * @param manifestName the name of the model manifest to wait for
     */
    public CompletableFuture<Void> waitForModelManifestActiveAsync(String
manifestName) {
        CompletableFuture<Void> result = new CompletableFuture<>();

        ScheduledExecutorService scheduler =
Executors.newSingleThreadScheduledExecutor();
        AtomicInteger secondsElapsed = new AtomicInteger(0);
        AtomicReference<ManifestStatus> lastStatus = new
AtomicReference<>(ManifestStatus.DRAFT);
```

```
        logger.info("Elapsed: 0s | Status: DRAFT");

        final Runnable pollTask = new Runnable() {
            @Override
            public void run() {
                int elapsed = secondsElapsed.incrementAndGet();

                // Only check status every 5 seconds
                if (elapsed % 5 == 0) {
                    GetModelManifestRequest request =
GetModelManifestRequest.builder()
                            .name(manifestName)
                            .build();

                    getAsyncClient().getModelManifest(request)
                            .whenComplete((response, exception) -> {
                                if (exception != null) {
                                    Throwable cause = exception instanceof
CompletionException ? exception.getCause() : exception;

                                    scheduler.shutdown();
                                    if (cause instanceof ResourceNotFoundException)
{
                                        result.completeExceptionally(new
RuntimeException("Model manifest not found: " + cause.getMessage(), cause));
                                    } else {
                                        result.completeExceptionally(new
RuntimeException("Error while polling model manifest status: " +
exception.getMessage(), exception));
                                    }
                                    return;
                                }

                                ManifestStatus status = response.status();
                                lastStatus.set(status);

                                if (status == ManifestStatus.ACTIVE) {
                                    logger.info("\rElapsed: {}s | Status: ACTIVE",
elapsed);

                                    scheduler.shutdown();
                                    result.complete(null);
                                } else if (status == ManifestStatus.INVALID) {
```

```
                                    logger.info("\rElapsed: {}s | Status: INVALID",
elapsed);
                                    scheduler.shutdown();
                                    result.completeExceptionally(new
RuntimeException("Model manifest became INVALID. Cannot proceed."));
                                } else {
                                    logger.info("\rElapsed: {}s | Status: {}",
elapsed, status);
                                }
                            });
                } else {
                    logger.info("\rElapsed: {}s | Status: {}", elapsed,
lastStatus.get());
                }
            }
        };

        // Start the task with an initial delay of 1 second, and repeat every second
        scheduler.scheduleAtFixedRate(pollTask, 1, 1, TimeUnit.SECONDS);
        return result;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 getModelManifest를 참조하세요.

`getVehicle`

다음 코드 예시는 `getVehicle`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Fetches the details of a vehicle.
     *
```

```
    * @param vehicleName the name of the vehicle to fetch details for
    * @return a {@link CompletableFuture} that completes when the vehicle details
have been fetched
    */
   public CompletableFuture<Void> getVehicleDetailsAsync(String vehicleName) {
       GetVehicleRequest request = GetVehicleRequest.builder()
               .vehicleName(vehicleName)
               .build();

       CompletableFuture<Void> result = new CompletableFuture<>();

       getAsyncClient().getVehicle(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception instanceof CompletionException ?
exception.getCause() : exception;

                       if (cause instanceof ResourceNotFoundException) {
                           result.completeExceptionally(cause); // don't rewrap
                       } else {
                           result.completeExceptionally(new
RuntimeException("Failed to fetch vehicle details: " + cause.getMessage(), cause));
                       }
                   } else {
                       Map<String, Object> details = new HashMap<>();
                       details.put("vehicleName", response.vehicleName());
                       details.put("arn", response.arn());
                       details.put("modelManifestArn",
response.modelManifestArn());
                       details.put("decoderManifestArn",
response.decoderManifestArn());
                       details.put("attributes", response.attributes());
                       details.put("creationTime",
response.creationTime().toString());
                       details.put("lastModificationTime",
response.lastModificationTime().toString());

                       logger.info("Vehicle Details:");
                       details.forEach((key, value) -> logger.info("• {} : {}",
key, value));

                       result.complete(null); // mark as successful
                   }
               });
```

```
        return result;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 getVehicle을 참조하세요.

`listSignalCatalogNodes`

다음 코드 예시는 `listSignalCatalogNodes`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Asynchronously retrieves a list of all nodes in the specified signal catalog.
     *
     * @param signalCatalogName the name of the signal catalog to retrieve nodes for
     * @return a {@link CompletableFuture} that, when completed, contains a {@link
List} of {@link Node} objects
     * representing all the nodes in the specified signal catalog
     */
    public CompletableFuture<List<Node>> listSignalCatalogNodeAsync(String
signalCatalogName) {
        ListSignalCatalogNodesRequest request =
ListSignalCatalogNodesRequest.builder()
                .name(signalCatalogName)
                .build();

        List<Node> allNodes = new ArrayList<>();

        return getAsyncClient().listSignalCatalogNodesPaginator(request)
                .subscribe(response -> allNodes.addAll(response.nodes()))
                .thenApply(v -> allNodes);
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 listSignalCatalogNodes를 참조하세요.

`updateDecoderManifest`

다음 코드 예시는 `updateDecoderManifest`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Updates the decoder manifest with the given name.
    *
    * @param name the name of the decoder manifest to update
    * @return a {@link CompletableFuture} that completes when the update operation
is finished
    */
   public CompletableFuture<Void> updateDecoderManifestAsync(String name) {
       UpdateDecoderManifestRequest request =
UpdateDecoderManifestRequest.builder()
               .name(name)
               .status(ManifestStatus.ACTIVE)
               .build();

       return getAsyncClient().updateDecoderManifest(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       throw new CompletionException("Failed to update decoder
manifest: " + exception.getMessage(), exception);
                   }
               })
               .thenApply(response -> null);
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 updateDecoderManifest를 참조하세요.

`updateModelManifest`

다음 코드 예시는 `updateModelManifest`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Updates the model manifest.
     *
     * @param name the name of the model manifest to update
     */
    public void updateModelManifestAsync(String name) {
        UpdateModelManifestRequest request = UpdateModelManifestRequest.builder()
                .name(name)
                .status(ManifestStatus.ACTIVE)
                .build();

        getAsyncClient().updateModelManifest(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        throw new CompletionException("Failed to update model
manifest: " + exception.getMessage(), exception);
                    }
                })
                .thenApply(response -> null);
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 updateModelManifest를 참조하세요.

AWS IoT SiteWise SDK for Java 2.x를 사용한 예제

다음 코드 예제에서는와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS IoT SiteWise.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시작하기

안녕하세요 AWS IoT SiteWise

다음 코드 예제에서는 AWS IoT SiteWise를 사용하여 시작하는 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
public class HelloSitewise {
    private static final Logger logger =
 LoggerFactory.getLogger(HelloSitewise.class);
    public static void main(String[] args) {
         fetchAssetModels();
    }
```

```
    /**
     * Fetches asset models using the provided {@link IoTSiteWiseAsyncClient}.
     */
    public static void fetchAssetModels() {
        IoTSiteWiseAsyncClient siteWiseAsyncClient =
 IoTSiteWiseAsyncClient.create();
        ListAssetModelsRequest assetModelsRequest = ListAssetModelsRequest.builder()
            .assetModelTypes(AssetModelType.ASSET_MODEL)
            .build();

        // Asynchronous paginator - process paginated results.
        ListAssetModelsPublisher listModelsPaginator =
 siteWiseAsyncClient.listAssetModelsPaginator(assetModelsRequest);
        CompletableFuture<Void> future = listModelsPaginator.subscribe(response -> {
            response.assetModelSummaries().forEach(assetSummary ->
                logger.info("Asset Model Name: {} ", assetSummary.name())
            );
        });

        // Wait for the asynchronous operation to complete
        future.join();
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListAssetModels을 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- AWS IoT SiteWise 자산 모델을 생성합니다.
- AWS IoT SiteWise 자산을 생성합니다.
- 속성 ID 값을 검색합니다.
- AWS IoT SiteWise 애셋으로 데이터를 전송합니다.
- AWS IoT SiteWise Asset 속성의 값을 검색합니다.
- AWS IoT SiteWise 포털을 생성합니다.

- AWS IoT SiteWise 게이트웨이를 생성합니다.
- AWS IoT SiteWise 게이트웨이를 설명합니다.
- AWS IoT SiteWise 자산을 삭제합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

AWS IoT SiteWise 기능을 보여주는 대화형 시나리오를 실행합니다.

```
public class SitewiseScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    private static final Logger logger =
 LoggerFactory.getLogger(SitewiseScenario.class);
    static Scanner scanner = new Scanner(System.in);

    private static final String ROLES_STACK = "RoleSitewise";

    static SitewiseActions sitewiseActions = new SitewiseActions();

    public static void main(String[] args) throws Throwable {
        Scanner scanner = new Scanner(System.in);
        String contactEmail = "user@mydomain.com"; // Change email address.
        String assetModelName = "MyAssetModel1";
        String assetName = "MyAsset1" ;
        String portalName = "MyPortal1" ;
        String gatewayName = "MyGateway1" ;
        String myThing =  "MyThing1" ;

        logger.info("""
            AWS IoT SiteWise is a fully managed software-as-a-service (SaaS) that
            makes it easy to collect, store, organize, and monitor data from
 industrial equipment and processes.
            It is designed to help industrial and manufacturing organizations
 collect data from their equipment and
```

```
            processes, and use that data to make informed decisions about their
operations.

            One of the key features of AWS IoT SiteWise is its ability to connect to
a wide range of industrial
            equipment and systems, including programmable logic controllers (PLCs),
sensors, and other
            industrial devices. It can collect data from these devices and organize
it into a unified data model,
            making it easier to analyze and gain insights from the data. AWS IoT
SiteWise also provides tools for
            visualizing the data, setting up alarms and alerts, and generating
reports.

            Another key feature of AWS IoT SiteWise is its ability to scale to
handle large volumes of data.
            It can collect and store data from thousands of devices and process
millions of data points per second,
            making it suitable for large-scale industrial operations. Additionally,
AWS IoT SiteWise is designed
            to be secure and compliant, with features like role-based access
controls, data encryption,
            and integration with other AWS services for additional security and
compliance features.

            Let's get started...
            """);

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        try {
            runScenario(assetModelName, assetName, portalName, contactEmail,
gatewayName, myThing);
        } catch (RuntimeException e) {
           logger.info(e.getMessage());
        }
    }

    public static void runScenario(String assetModelName, String assetName,  String
portalName, String contactEmail, String gatewayName, String myThing) throws
Throwable {
        logger.info("Use AWS CloudFormation to create an IAM role that is required
for this scenario.");
```

```
        CloudFormationHelper.deployCloudFormationStack(ROLES_STACK);
        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputsAsync(ROLES_STACK).join();
        String iamRole = stackOutputs.get("SitewiseRoleArn");
        logger.info("The ARN of the IAM role is {}",iamRole);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("1. Create an AWS SiteWise Asset Model");
        logger.info("""
             An AWS IoT SiteWise Asset Model is a way to represent the physical
assets, such as equipment,
             processes, and systems, that exist in an industrial environment. This
model provides a structured and
             hierarchical representation of these assets, allowing users to define
the relationships and properties
             of each asset.

             This scenario creates two asset model properties: temperature and
humidity.
           """);
        waitForInputToContinue(scanner);
        String assetModelId = null;
        try {
            CreateAssetModelResponse response =
sitewiseActions.createAssetModelAsync(assetModelName).join();
            assetModelId = response.assetModelId();
            logger.info("Asset Model successfully created. Asset Model ID: {}. ",
assetModelId);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceAlreadyExistsException) {
                try {
                    assetModelId =
sitewiseActions.getAssetModelIdAsync(assetModelName).join();
                    logger.info("The Asset Model {} already exists. The id of the
existing model is {}. Moving on...", assetModelName, assetModelId);
                } catch (CompletionException cex) {
                    logger.error("Exception thrown acquiring the asset model id:
{}", cex.getCause().getCause(), cex);
                    return;
                }
            } else {
```

```
                logger.info("An unexpected error occurred: " + cause.getMessage(),
cause);
                return;
            }
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("2. Create an AWS IoT SiteWise Asset");
        logger.info("""
             The IoT SiteWise model that we just created defines the structure and
metadata for your physical assets.
             Now we create an asset from the asset model.

            """);
        logger.info("Let's wait 30 seconds for the asset to be ready.");
        countdown(30);
        waitForInputToContinue(scanner);
        String assetId;
        try {
            CreateAssetResponse response =
sitewiseActions.createAssetAsync(assetName, assetModelId).join();
            assetId = response.assetId();
            logger.info("Asset created with ID: {}", assetId);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The asset model id was not found: {}",
cause.getMessage(), cause);
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Retrieve the property ID values");
        logger.info("""
             To send data to an asset, we need to get the property ID values. In
this scenario, we access the
             temperature and humidity property ID values.
```

```
            """);
        waitForInputToContinue(scanner);
        Map<String, String>  propertyIds = null;
        try {
            propertyIds = sitewiseActions.getPropertyIds(assetModelId).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IoTSiteWiseException) {
                logger.error("IoTSiteWiseException occurred: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
            return;
        }
        String humPropId =  propertyIds.get("Humidity");
        logger.info("The Humidity property Id is {}", humPropId);
        String tempPropId = propertyIds.get("Temperature");
        logger.info("The Temperature property Id is {}", tempPropId);

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Send data to an AWS IoT SiteWise Asset");
        logger.info("""
            By sending data to an IoT SiteWise Asset, you can aggregate data from
            multiple sources, normalize the data into a standard format, and store
it in a
            centralized location. This makes it easier to analyze and gain insights
from the data.

            In this example, we generate sample temperature and humidity data and
send it to the AWS IoT SiteWise asset.

            """);
        waitForInputToContinue(scanner);
        try {
            sitewiseActions.sendDataToSiteWiseAsync(assetId, tempPropId,
humPropId).join();
            logger.info("Data sent successfully.");
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
```

```
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The AWS resource was not found: {}",
cause.getMessage(), cause);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Retrieve the value of the IoT SiteWise Asset property");
        logger.info("""
            IoT SiteWise is an AWS service that allows you to collect, process, and
analyze industrial data
            from connected equipment and sensors. One of the key benefits of reading
an IoT SiteWise property
            is the ability to gain valuable insights from your industrial data.

            """);
        waitForInputToContinue(scanner);
        try {
            Double assetVal = sitewiseActions.getAssetPropValueAsync(tempPropId,
assetId).join();
            logger.info("The property name is: {}", "Temperature");
            logger.info("The value of this property is: {}", assetVal);

            waitForInputToContinue(scanner);

            assetVal = sitewiseActions.getAssetPropValueAsync(humPropId,
assetId).join();
            logger.info("The property name is: {}", "Humidity");
            logger.info("The value of this property is: {}", assetVal);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    logger.info("The AWS resource was not found: {}",
cause.getMessage(), cause);
                } else {
                    logger.info("An unexpected error occurred: {}",
cause.getMessage(), cause);
                }
```

```
                return;
            }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. Create an IoT SiteWise Gateway");
        logger.info(
            """
                IoT SiteWise Gateway serves as the bridge between industrial
equipment, sensors, and the
                cloud-based IoT SiteWise service. It is responsible for securely
collecting, processing, and
                transmitting data from various industrial assets to the IoT SiteWise
platform,
                enabling real-time monitoring, analysis, and optimization of
industrial operations.

                """);
        waitForInputToContinue(scanner);
        String gatewayId = "";
        try {
            gatewayId = sitewiseActions.createGatewayAsync(gatewayName,
myThing).join();
            logger.info("Gateway creation completed successfully. id is {}",
gatewayId );
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IoTSiteWiseException siteWiseEx) {
                logger.error("IoT SiteWise error occurred: Error message: {}, Error
code {}",
                        siteWiseEx.getMessage(),
siteWiseEx.awsErrorDetails().errorCode(), siteWiseEx);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage());
            }
            return;
        }
        logger.info(DASHES);
        logger.info(DASHES);

        logger.info("7. Describe the IoT SiteWise Gateway");
         waitForInputToContinue(scanner);
```

```
        try {
            sitewiseActions.describeGatewayAsync(gatewayId)
                .thenAccept(response -> {
                    logger.info("Gateway Name: {}", response.gatewayName());
                    logger.info("Gateway ARN: {}", response.gatewayArn());
                    logger.info("Gateway Platform: {}", response.gatewayPlatform());
                    logger.info("Gateway Creation Date: {}",
response.creationDate());
                }).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException notFoundException) {
                logger.error("A ResourceNotFoundException occurred: Error message:
{}, Error code {}",
                        notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Delete the AWS IoT SiteWise Assets");
        logger.info(
            """
            Before you can delete the Asset Model, you must delete the assets.

            """);
        logger.info("Would you like to delete the IoT SiteWise Assets? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete the SiteWise assets.");

            try {
                sitewiseActions.deleteGatewayAsync(gatewayId).join();
                logger.info("Gateway {} was deleted successfully.", gatewayId);
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException notFoundException) {
                    logger.error("A ResourceNotFoundException occurred: Error
message: {}, Error code {}",
```

```
                            notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage());
                }
            }

            try {
                sitewiseActions.deleteAssetAsync(assetId).join();
                logger.info("Request to delete asset {} sent successfully",
assetId);
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException notFoundException) {
                    logger.error("A ResourceNotFoundException occurred: Error
message: {}, Error code {}",
                            notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage());
                }
            }
            logger.info("Let's wait 1 minute for the asset to be deleted.");
            countdown(60);
            waitForInputToContinue(scanner);
            logger.info("Delete the AWS IoT SiteWise Asset Model");
            try {
                sitewiseActions.deleteAssetModelAsync(assetModelId).join();
                logger.info("Asset model deleted successfully.");
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException notFoundException) {
                    logger.error("A ResourceNotFoundException occurred: Error
message: {}, Error code {}",
                            notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage());
                }
            }
            waitForInputToContinue(scanner);
```

```
        } else {
            logger.info("The resources will not be deleted.");
        }
        logger.info(DASHES);

        logger.info(DASHES);
        CloudFormationHelper.destroyCloudFormationStack(ROLES_STACK);
        logger.info("This concludes the AWS IoT SiteWise Scenario");
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }

    public static void countdown(int totalSeconds) throws InterruptedException {
        for (int i = totalSeconds; i >= 0; i--) {
            int displayMinutes = i / 60;
            int displaySeconds = i % 60;
            System.out.printf("\r%02d:%02d", displayMinutes, displaySeconds);
            Thread.sleep(1000); // Wait for 1 second
        }
        System.out.println(); // Move to the next line after countdown
        logger.info("Countdown complete!");
    }
}
```

AWS IoT SiteWise SDK 메서드의 래퍼 클래스입니다.

```
public class SitewiseActions {

   private static final Logger logger =
 LoggerFactory.getLogger(SitewiseActions.class);

   private static IoTSiteWiseAsyncClient ioTSiteWiseAsyncClient;

   private static IoTSiteWiseAsyncClient getAsyncClient() {
       if (ioTSiteWiseAsyncClient == null) {
           SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
               .maxConcurrency(100)
               .connectionTimeout(Duration.ofSeconds(60))
               .readTimeout(Duration.ofSeconds(60))
               .writeTimeout(Duration.ofSeconds(60))
               .build();

           ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
               .apiCallTimeout(Duration.ofMinutes(2))
               .apiCallAttemptTimeout(Duration.ofSeconds(90))
               .retryStrategy(RetryMode.STANDARD)
               .build();

           ioTSiteWiseAsyncClient = IoTSiteWiseAsyncClient.builder()
               .httpClient(httpClient)
               .overrideConfiguration(overrideConfig)
               .build();
       }
       return ioTSiteWiseAsyncClient;
   }


   /**
    * Creates an asset model.
    *
    * @param name the name of the asset model to create.
    * @return a {@link CompletableFuture} that represents a {@link
 CreateAssetModelResponse} result. The calling code
    *         can attach callbacks, then handle the result or exception by calling
 {@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
```

```
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
    *         available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<CreateAssetModelResponse> createAssetModelAsync(String
name) {
       PropertyType humidity = PropertyType.builder()
           .measurement(Measurement.builder().build())
           .build();

       PropertyType temperaturePropertyType = PropertyType.builder()
           .measurement(Measurement.builder().build())
           .build();

       AssetModelPropertyDefinition temperatureProperty =
AssetModelPropertyDefinition.builder()
           .name("Temperature")
           .dataType(PropertyDataType.DOUBLE)
           .type(temperaturePropertyType)
           .build();

       AssetModelPropertyDefinition humidityProperty =
AssetModelPropertyDefinition.builder()
           .name("Humidity")
           .dataType(PropertyDataType.DOUBLE)
           .type(humidity)
           .build();

       CreateAssetModelRequest createAssetModelRequest =
CreateAssetModelRequest.builder()
           .assetModelName(name)
           .assetModelDescription("This is my asset model")
           .assetModelProperties(temperatureProperty, humidityProperty)
           .build();

       return getAsyncClient().createAssetModel(createAssetModelRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("Failed to create asset model: {} ",
exception.getCause().getMessage());
               }
```

```
            });
    }


    /**
     * Creates an asset with the specified name and asset model Id.
     *
     * @param assetName    the name of the asset to create.
     * @param assetModelId the Id of the asset model to associate with the asset.
     * @return a {@link CompletableFuture} that represents a {@link
CreateAssetResponse} result. The calling code can
     *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
     *         available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<CreateAssetResponse> createAssetAsync(String assetName,
String assetModelId) {
        CreateAssetRequest createAssetRequest = CreateAssetRequest.builder()
            .assetModelId(assetModelId)
            .assetDescription("Created using the AWS SDK for Java")
            .assetName(assetName)
            .build();

        return getAsyncClient().createAsset(createAssetRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to create asset: {}",
exception.getCause().getMessage());
                }
            });
    }

    /**
     * Sends data to the SiteWise service.
     *
     * @param assetId        the ID of the asset to which the data will be sent.
     * @param tempPropertyId the ID of the temperature property.
```

```
    * @param humidityPropId the ID of the humidity property.
    * @return a {@link CompletableFuture} that represents a {@link
BatchPutAssetPropertyValueResponse} result. The
    *         calling code can attach callbacks, then handle the result or
exception by calling
    *         {@link CompletableFuture#join()} or {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
    *         available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<BatchPutAssetPropertyValueResponse>
sendDataToSiteWiseAsync(String assetId, String tempPropertyId, String
humidityPropId) {
       Map<String, Double> sampleData = generateSampleData();
       long timestamp = Instant.now().toEpochMilli();

       TimeInNanos time = TimeInNanos.builder()
           .timeInSeconds(timestamp / 1000)
           .offsetInNanos((int) ((timestamp % 1000) * 1000000))
           .build();

       BatchPutAssetPropertyValueRequest request =
BatchPutAssetPropertyValueRequest.builder()
           .entries(Arrays.asList(
               PutAssetPropertyValueEntry.builder()
                   .entryId("entry-3")
                   .assetId(assetId)
                   .propertyId(tempPropertyId)
                   .propertyValues(Arrays.asList(
                       AssetPropertyValue.builder()
                           .value(Variant.builder()
                               .doubleValue(sampleData.get("Temperature"))
                               .build())
                           .timestamp(time)
                           .build()
                   ))
                   .build(),
               PutAssetPropertyValueEntry.builder()
                   .entryId("entry-4")
                   .assetId(assetId)
```

```
                    .propertyId(humidityPropId)
                    .propertyValues(Arrays.asList(
                        AssetPropertyValue.builder()
                            .value(Variant.builder()
                                .doubleValue(sampleData.get("Humidity"))
                                .build())
                            .timestamp(time)
                            .build()
                    ))
                    .build()
            ))
            .build();

        return getAsyncClient().batchPutAssetPropertyValue(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An exception occurred: {}",
exception.getCause().getMessage());
                }
            });
    }

    /**
     * Fetches the value of an asset property.
     *
     * @param propId  the ID of the asset property to fetch.
     * @param assetId the ID of the asset to fetch the property value for.
     * @return a {@link CompletableFuture} that represents a {@link Double} result.
The calling code can attach
     *         callbacks, then handle the result or exception by calling {@link
CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<Double> getAssetPropValueAsync(String propId, String
assetId) {
        GetAssetPropertyValueRequest assetPropertyValueRequest =
GetAssetPropertyValueRequest.builder()
```

```
                .propertyId(propId)
                .assetId(assetId)
                .build();

        return getAsyncClient().getAssetPropertyValue(assetPropertyValueRequest)
                .handle((response, exception) -> {
                    if (exception != null) {
                        logger.error("Error occurred while fetching property value:
{}.", exception.getCause().getMessage());
                        throw (CompletionException) exception;
                    }
                    return response.propertyValue().value().doubleValue();
                });
    }

    /**
     * Retrieves the property IDs associated with a specific asset model.
     *
     * @param assetModelId the ID of the asset model that defines the properties.
     * @return a {@link CompletableFuture} that represents a {@link Map} result that
associates the property name to the
     *         propert ID. The calling code can attach callbacks, then handle the
result or exception by calling
     *         {@link CompletableFuture#join()} or {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<Map<String, String>> getPropertyIds(String
assetModelId) {
        ListAssetModelPropertiesRequest modelPropertiesRequest =
ListAssetModelPropertiesRequest.builder().assetModelId(assetModelId).build();
        return getAsyncClient().listAssetModelProperties(modelPropertiesRequest)
            .handle((response, throwable) -> {
                if (response != null) {
                    return response.assetModelPropertySummaries().stream()
                        .collect(Collectors
                            .toMap(AssetModelPropertySummary::name,
AssetModelPropertySummary::id));
                } else {
```

```
                    logger.error("Error occurred while fetching property IDs: {}.",
throwable.getCause().getMessage());
                    throw (CompletionException) throwable;
                }
            });
    }

    /**
     * Deletes an asset.
     *
     * @param assetId the ID of the asset to be deleted.
     * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetResponse} result. The calling code can
     *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<DeleteAssetResponse> deleteAssetAsync(String assetId) {
        DeleteAssetRequest deleteAssetRequest = DeleteAssetRequest.builder()
            .assetId(assetId)
            .build();

        return getAsyncClient().deleteAsset(deleteAssetRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An error occurred deleting asset with id: {}",
assetId);
                }
            });
    }

    /**
     * Deletes an Asset Model with the specified ID.
     *
     * @param assetModelId the ID of the Asset Model to delete.
     * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetModelResponse} result. The calling code
```

```
    *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<DeleteAssetModelResponse> deleteAssetModelAsync(String
assetModelId) {
       DeleteAssetModelRequest deleteAssetModelRequest =
DeleteAssetModelRequest.builder()
           .assetModelId(assetModelId)
           .build();

       return getAsyncClient().deleteAssetModel(deleteAssetModelRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("Failed to delete asset model with ID:{}.",
exception.getMessage());
               }
           });
   }

   /**
    * Retrieves the asset model ID for the given asset model name.
    *
    * @param assetModelName the name of the asset model for the ID.
    * @return a {@link CompletableFuture} that represents a {@link String} result
of the asset model ID or null if the
    *         asset model cannot be found. The calling code can attach callbacks,
then handle the result or exception
    *         by calling {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
```

```
     */
    public CompletableFuture<String> getAssetModelIdAsync(String assetModelName) {
        ListAssetModelsRequest listAssetModelsRequest =
ListAssetModelsRequest.builder().build();
        return getAsyncClient().listAssetModels(listAssetModelsRequest)
                .handle((listAssetModelsResponse, exception) -> {
                    if (exception != null) {
                        logger.error("Failed to retrieve Asset Model ID: {}",
exception.getCause().getMessage());
                        throw (CompletionException) exception;
                    }
                    for (AssetModelSummary assetModelSummary :
listAssetModelsResponse.assetModelSummaries()) {
                        if (assetModelSummary.name().equals(assetModelName)) {
                            return assetModelSummary.id();
                        }
                    }
                    return null;
                });
    }


    /**
     * Creates a new IoT Sitewise gateway.
     *
     * @param gatewayName The name of the gateway to create.
     * @param myThing     The name of the core device thing to associate with the
gateway.
     * @return a {@link CompletableFuture} that represents a {@link String} result
of the gateways ID. The calling code
     *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<String> createGatewayAsync(String gatewayName, String
myThing) {
        GreengrassV2 gg = GreengrassV2.builder()
```

```
            .coreDeviceThingName(myThing)
            .build();

        GatewayPlatform platform = GatewayPlatform.builder()
            .greengrassV2(gg)
            .build();

        Map<String, String> tag = new HashMap<>();
        tag.put("Environment", "Production");

        CreateGatewayRequest createGatewayRequest = CreateGatewayRequest.builder()
            .gatewayName(gatewayName)
            .gatewayPlatform(platform)
            .tags(tag)
            .build();

        return getAsyncClient().createGateway(createGatewayRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    logger.error("Error creating the gateway.");
                    throw (CompletionException) exception;
                }
                logger.info("The ARN of the gateway is {}" ,
response.gatewayArn());
                return response.gatewayId();
            });
    }

    /**
     * Deletes the specified gateway.
     *
     * @param gatewayId the ID of the gateway to delete.
     * @return a {@link CompletableFuture} that represents a {@link
DeleteGatewayResponse} result.. The calling code
     *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
```

```
     */
    public CompletableFuture<DeleteGatewayResponse> deleteGatewayAsync(String
gatewayId) {
        DeleteGatewayRequest deleteGatewayRequest = DeleteGatewayRequest.builder()
            .gatewayId(gatewayId)
            .build();

        return getAsyncClient().deleteGateway(deleteGatewayRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to delete gateway: {}",
exception.getCause().getMessage());
                }
            });
    }

    /**
     * Describes the specified gateway.
     *
     * @param gatewayId the ID of the gateway to describe.
     * @return a {@link CompletableFuture} that represents a {@link
DescribeGatewayResponse} result. The calling code
     *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<DescribeGatewayResponse> describeGatewayAsync(String
gatewayId) {
        DescribeGatewayRequest request = DescribeGatewayRequest.builder()
            .gatewayId(gatewayId)
            .build();

        return getAsyncClient().describeGateway(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An error occurred during the describeGateway
method: {}", exception.getCause().getMessage());
```

```
                }
            });
    }

    private static Map<String, Double> generateSampleData() {
        Map<String, Double> data = new HashMap<>();
        data.put("Temperature", 23.5);
        data.put("Humidity", 65.0);
        return data;
    }
}
```

작업

`BatchPutAssetPropertyValue`

다음 코드 예시는 `BatchPutAssetPropertyValue`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Sends data to the SiteWise service.
     *
     * @param assetId        the ID of the asset to which the data will be sent.
     * @param tempPropertyId the ID of the temperature property.
     * @param humidityPropId the ID of the humidity property.
     * @return a {@link CompletableFuture} that represents a {@link
BatchPutAssetPropertyValueResponse} result. The
     *         calling code can attach callbacks, then handle the result or
exception by calling
     *         {@link CompletableFuture#join()} or {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
```

```
    *         available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<BatchPutAssetPropertyValueResponse>
sendDataToSiteWiseAsync(String assetId, String tempPropertyId, String
humidityPropId) {
       Map<String, Double> sampleData = generateSampleData();
       long timestamp = Instant.now().toEpochMilli();

       TimeInNanos time = TimeInNanos.builder()
           .timeInSeconds(timestamp / 1000)
           .offsetInNanos((int) ((timestamp % 1000) * 1000000))
           .build();

       BatchPutAssetPropertyValueRequest request =
BatchPutAssetPropertyValueRequest.builder()
           .entries(Arrays.asList(
               PutAssetPropertyValueEntry.builder()
                   .entryId("entry-3")
                   .assetId(assetId)
                   .propertyId(tempPropertyId)
                   .propertyValues(Arrays.asList(
                       AssetPropertyValue.builder()
                           .value(Variant.builder()
                               .doubleValue(sampleData.get("Temperature"))
                               .build())
                           .timestamp(time)
                           .build()
                   ))
                   .build(),
               PutAssetPropertyValueEntry.builder()
                   .entryId("entry-4")
                   .assetId(assetId)
                   .propertyId(humidityPropId)
                   .propertyValues(Arrays.asList(
                       AssetPropertyValue.builder()
                           .value(Variant.builder()
                               .doubleValue(sampleData.get("Humidity"))
                               .build())
                           .timestamp(time)
                           .build()
                   ))
```

```
                    .build()
            ))
            .build();

        return getAsyncClient().batchPutAssetPropertyValue(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An exception occurred: {}",
exception.getCause().getMessage());
                }
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 BatchPutAssetPropertyValue를 참조하세요.

`CreateAsset`

다음 코드 예시는 `CreateAsset`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Creates an asset with the specified name and asset model Id.
     *
     * @param assetName    the name of the asset to create.
     * @param assetModelId the Id of the asset model to associate with the asset.
     * @return a {@link CompletableFuture} that represents a {@link
CreateAssetResponse} result. The calling code can
     *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
```

```
     *         available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<CreateAssetResponse> createAssetAsync(String assetName,
String assetModelId) {
        CreateAssetRequest createAssetRequest = CreateAssetRequest.builder()
            .assetModelId(assetModelId)
            .assetDescription("Created using the AWS SDK for Java")
            .assetName(assetName)
            .build();

        return getAsyncClient().createAsset(createAssetRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to create asset: {}",
exception.getCause().getMessage());
                }
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateAsset을 참조하세요.

`CreateAssetModel`

다음 코드 예시는 `CreateAssetModel`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Creates an asset model.
     *
     * @param name the name of the asset model to create.
```

```
    * @return a {@link CompletableFuture} that represents a {@link
CreateAssetModelResponse} result. The calling code
    *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
    *         available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<CreateAssetModelResponse> createAssetModelAsync(String
name) {
       PropertyType humidity = PropertyType.builder()
           .measurement(Measurement.builder().build())
           .build();

       PropertyType temperaturePropertyType = PropertyType.builder()
           .measurement(Measurement.builder().build())
           .build();

       AssetModelPropertyDefinition temperatureProperty =
AssetModelPropertyDefinition.builder()
           .name("Temperature")
           .dataType(PropertyDataType.DOUBLE)
           .type(temperaturePropertyType)
           .build();

       AssetModelPropertyDefinition humidityProperty =
AssetModelPropertyDefinition.builder()
           .name("Humidity")
           .dataType(PropertyDataType.DOUBLE)
           .type(humidity)
           .build();

       CreateAssetModelRequest createAssetModelRequest =
CreateAssetModelRequest.builder()
           .assetModelName(name)
           .assetModelDescription("This is my asset model")
           .assetModelProperties(temperatureProperty, humidityProperty)
           .build();
```

```
        return getAsyncClient().createAssetModel(createAssetModelRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to create asset model: {} ",
exception.getCause().getMessage());
                }
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateAssetModel을 참조하세요.

`CreateGateway`

다음 코드 예시는 `CreateGateway`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Creates a new IoT Sitewise gateway.
     *
     * @param gatewayName The name of the gateway to create.
     * @param myThing     The name of the core device thing to associate with the
gateway.
     * @return a {@link CompletableFuture} that represents a {@link String} result
of the gateways ID. The calling code
     *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
```

```
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<String> createGatewayAsync(String gatewayName, String
myThing) {
        GreengrassV2 gg = GreengrassV2.builder()
            .coreDeviceThingName(myThing)
            .build();

        GatewayPlatform platform = GatewayPlatform.builder()
            .greengrassV2(gg)
            .build();

        Map<String, String> tag = new HashMap<>();
        tag.put("Environment", "Production");

        CreateGatewayRequest createGatewayRequest = CreateGatewayRequest.builder()
            .gatewayName(gatewayName)
            .gatewayPlatform(platform)
            .tags(tag)
            .build();

        return getAsyncClient().createGateway(createGatewayRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    logger.error("Error creating the gateway.");
                    throw (CompletionException) exception;
                }
                logger.info("The ARN of the gateway is {}" ,
response.gatewayArn());
                return response.gatewayId();
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateGateway를 참조하세요.

`DeleteAsset`

다음 코드 예시는 `DeleteAsset`의 사용 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Deletes an asset.
     *
     * @param assetId the ID of the asset to be deleted.
     * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetResponse} result. The calling code can
     *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<DeleteAssetResponse> deleteAssetAsync(String assetId) {
        DeleteAssetRequest deleteAssetRequest = DeleteAssetRequest.builder()
            .assetId(assetId)
            .build();

        return getAsyncClient().deleteAsset(deleteAssetRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An error occurred deleting asset with id: {}",
assetId);
                }
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteAsset을 참조하세요.

`DeleteAssetModel`

다음 코드 예시는 `DeleteAssetModel`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Deletes an Asset Model with the specified ID.
    *
    * @param assetModelId the ID of the Asset Model to delete.
    * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetModelResponse} result. The calling code
    *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<DeleteAssetModelResponse> deleteAssetModelAsync(String
assetModelId) {
       DeleteAssetModelRequest deleteAssetModelRequest =
DeleteAssetModelRequest.builder()
           .assetModelId(assetModelId)
           .build();

       return getAsyncClient().deleteAssetModel(deleteAssetModelRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("Failed to delete asset model with ID:{}.",
exception.getMessage());
               }
           });
```

```
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteAssetModel을 참조하세요.

`DeleteGateway`

다음 코드 예시는 `DeleteGateway`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Deletes the specified gateway.
    *
    * @param gatewayId the ID of the gateway to delete.
    * @return a {@link CompletableFuture} that represents a {@link
DeleteGatewayResponse} result.. The calling code
    *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<DeleteGatewayResponse> deleteGatewayAsync(String
gatewayId) {
       DeleteGatewayRequest deleteGatewayRequest = DeleteGatewayRequest.builder()
           .gatewayId(gatewayId)
           .build();

       return getAsyncClient().deleteGateway(deleteGatewayRequest)
           .whenComplete((response, exception) -> {
```

```
                if (exception != null) {
                    logger.error("Failed to delete gateway: {}",
exception.getCause().getMessage());
                }
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteGateway를 참조하세요.

`DescribeAssetModel`

다음 코드 예시는 `DescribeAssetModel`의 사용 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Retrieves the property IDs associated with a specific asset model.
     *
     * @param assetModelId the ID of the asset model that defines the properties.
     * @return a {@link CompletableFuture} that represents a {@link Map} result that
associates the property name to the
     *         propert ID. The calling code can attach callbacks, then handle the
result or exception by calling
     *         {@link CompletableFuture#join()} or {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<Map<String, String>> getPropertyIds(String
assetModelId) {
```

```
        ListAssetModelPropertiesRequest modelPropertiesRequest =
ListAssetModelPropertiesRequest.builder().assetModelId(assetModelId).build();
        return getAsyncClient().listAssetModelProperties(modelPropertiesRequest)
            .handle((response, throwable) -> {
                if (response != null) {
                    return response.assetModelPropertySummaries().stream()
                        .collect(Collectors
                            .toMap(AssetModelPropertySummary::name,
AssetModelPropertySummary::id));
                } else {
                    logger.error("Error occurred while fetching property IDs: {}.",
throwable.getCause().getMessage());
                    throw (CompletionException) throwable;
                }
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeAssetModel을 참조하세요.

`DescribeGateway`

다음 코드 예시는 `DescribeGateway`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Describes the specified gateway.
     *
     * @param gatewayId the ID of the gateway to describe.
     * @return a {@link CompletableFuture} that represents a {@link
DescribeGatewayResponse} result. The calling code
     *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
```

```
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<DescribeGatewayResponse> describeGatewayAsync(String
gatewayId) {
        DescribeGatewayRequest request = DescribeGatewayRequest.builder()
            .gatewayId(gatewayId)
            .build();

        return getAsyncClient().describeGateway(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An error occurred during the describeGateway
method: {}", exception.getCause().getMessage());
                }
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeGateway를 참조하세요.

`GetAssetPropertyValue`

다음 코드 예시는 `GetAssetPropertyValue`의 사용 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Fetches the value of an asset property.
     *
```

```
    * @param propId  the ID of the asset property to fetch.
    * @param assetId the ID of the asset to fetch the property value for.
    * @return a {@link CompletableFuture} that represents a {@link Double} result.
The calling code can attach
    *         callbacks, then handle the result or exception by calling {@link
CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<Double> getAssetPropValueAsync(String propId, String
assetId) {
       GetAssetPropertyValueRequest assetPropertyValueRequest =
GetAssetPropertyValueRequest.builder()
               .propertyId(propId)
               .assetId(assetId)
               .build();

       return getAsyncClient().getAssetPropertyValue(assetPropertyValueRequest)
               .handle((response, exception) -> {
                   if (exception != null) {
                       logger.error("Error occurred while fetching property value:
{}.", exception.getCause().getMessage());
                       throw (CompletionException) exception;
                   }
                   return response.propertyValue().value().doubleValue();
               });
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetAssetPropertyValue를 참조하세요.

`ListAssetModels`

다음 코드 예시는 `ListAssetModels`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Retrieves the asset model ID for the given asset model name.
     *
     * @param assetModelName the name of the asset model for the ID.
     * @return a {@link CompletableFuture} that represents a {@link String} result
of the asset model ID or null if the
     *         asset model cannot be found. The calling code can attach callbacks,
then handle the result or exception
     *         by calling {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<String> getAssetModelIdAsync(String assetModelName) {
        ListAssetModelsRequest listAssetModelsRequest =
ListAssetModelsRequest.builder().build();
        return getAsyncClient().listAssetModels(listAssetModelsRequest)
                .handle((listAssetModelsResponse, exception) -> {
                    if (exception != null) {
                        logger.error("Failed to retrieve Asset Model ID: {}",
exception.getCause().getMessage());
                        throw (CompletionException) exception;
                    }
                    for (AssetModelSummary assetModelSummary :
listAssetModelsResponse.assetModelSummaries()) {
                        if (assetModelSummary.name().equals(assetModelName)) {
                            return assetModelSummary.id();
                        }
                    }
```

```
                    return null;
                });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListAssetModels을 참조하세요.

Java 2.x용 SDK를 사용하는 Amazon Keyspaces 예제

다음 코드 예제에서는 Amazon Keyspaces와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시작하기

Hello Amazon Keyspaces

다음 코드 예제에서는 Amazon Keyspaces 사용을 시작하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
```

```
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListKeyspaces를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 키스페이스와 테이블을 생성하세요. 테이블 스키마에는 영화 데이터가 저장되며 특정 시점으로 복구가 활성화되어 있습니다.
- SigV4 인증을 통한 보안 TLS 연결을 사용하여 키스페이스에 연결합니다.
- 테이블을 쿼리합니다. 영화 데이터를 추가, 검색 및 업데이트합니다.
- 테이블을 업데이트 하세요. 열을 추가하여 시청한 영화를 추적합니다.
- 테이블을 이전 상태로 복원하고 리소스를 정리합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including 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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 항목을 참조하세요.
  - CreateKeyspace
  - CreateTable
  - DeleteKeyspace
  - DeleteTable
  - GetKeyspace
  - GetTable
  - ListKeyspaces
  - ListTables
  - RestoreTable
  - UpdateTable

작업

`CreateKeyspace`

다음 코드 예시는 `CreateKeyspace`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateKeyspace를 참조하세요.

`CreateTable`

다음 코드 예시는 `CreateTable`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateTable을 참조하세요.

`DeleteKeyspace`

다음 코드 예시는 `DeleteKeyspace`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteKeyspace를 참조하세요.

`DeleteTable`

다음 코드 예시는 `DeleteTable`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteTable을 참조하세요.

`GetKeyspace`

다음 코드 예시는 `GetKeyspace`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 GetKeyspace를 참조하세요.

`GetTable`

다음 코드 예시는 `GetTable`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 GetTable을 참조하세요.

`ListKeyspaces`

다음 코드 예시는 `ListKeyspaces`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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);
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListKeyspaces를 참조하세요.

`ListTables`

다음 코드 예시는 `ListTables`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListTables를 참조하세요.

`RestoreTable`

다음 코드 예시는 `RestoreTable`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 RestoreTable을 참조하세요.

`UpdateTable`

다음 코드 예시는 `UpdateTable`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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);
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateTable을 참조하세요.

Java 2.x용 SDK를 사용하는 Kinesis 예제

다음 코드 예제에서는 Kinesis와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업
- 서버리스 예제

작업

`CreateStream`

다음 코드 예시는 `CreateStream`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateStream을 참조하세요.

`DeleteStream`

다음 코드 예시는 `DeleteStream`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteStream을 참조하세요.

`GetRecords`

다음 코드 예시는 `GetRecords`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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()));
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetRecords를 참조하세요.

`PutRecord`

다음 코드 예시는 `PutRecord`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 PutRecord를 참조하세요.

서버리스 예제

Kinesis 트리거에서 간접적으로 Lambda 함수 간접 호출

다음 코드 예제에서는 Kinesis 스트림에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 Kinesis 페이로드를 검색하고, Base64에서 디코딩하고, 레코드 콘텐츠를 로깅합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Java를 사용하여 Lambda에서 Kinesis 이벤트를 사용합니다.

```
// 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;
    }

}
```

Kinesis 트리거로 Lambda 함수에 대한 배치 항목 실패 보고

다음 코드 예제는 Kinesis 스트림에서 이벤트를 수신하는 Lambda 함수에 대한 부분 배치 응답을 구현하는 방법을 보여줍니다. 이 함수는 응답으로 배치 항목 실패를 보고하고 나중에 해당 메시지를 다시 시도하도록 Lambda에 신호를 보냅니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Java를 사용하여 Lambda로 Kinesis 배치 항목 실패 보고.

```
// 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 SDK for Java 2.x를 사용한 예제

다음 코드 예제에서는와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS KMS.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시작하기

안녕하세요 AWS KMS

다음 코드 예제에서는 AWS Key Management Service를 사용하여 시작하는 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.services.kms.KmsAsyncClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.paginators.ListKeysPublisher;
import java.util.concurrent.CompletableFuture;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * 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) {
        listAllKeys();
    }

    public static void listAllKeys() {
        KmsAsyncClient kmsAsyncClient = KmsAsyncClient.builder()
            .build();
        ListKeysRequest listKeysRequest = ListKeysRequest.builder()
            .limit(15)
            .build();

        /*
         * The `subscribe` method is required when using paginator methods in the
 AWS SDK
         * because paginator methods return an instance of a `ListKeysPublisher`,
 which is
         * based on a reactive stream. This allows asynchronous retrieval of
 paginated
         * results as they become available. By subscribing to the stream, we can
 process
         * each page of results as they are emitted.
         */
        ListKeysPublisher keysPublisher =
 kmsAsyncClient.listKeysPaginator(listKeysRequest);
        CompletableFuture<Void> future = keysPublisher
            .subscribe(r -> r.keys().forEach(key ->
                System.out.println("The key ARN is: " + key.keyArn() + ". The key Id
 is: " + key.keyId())))
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    System.err.println("Error occurred: " + exception.getMessage());
                } else {
                    System.out.println("Successfully listed all keys.");
                }
            });
```

```
        try {
            future.join();
        } catch (Exception e) {
            System.err.println("Failed to list keys: " + e.getMessage());
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListKeys를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- KMS 키를 생성합니다.
- 계정의 KMS 키를 나열하고 해당 키에 대한 세부 정보를 확인하세요.
- KMS 키를 활성화 및 비활성화합니다.
- 클라이언트 측 암호화에 사용할 수 있는 대칭 데이터 키를 생성하세요.
- 데이터에 디지털 방식으로 서명하는 데 사용되는 비대칭 키를 생성합니다.
- 키에 태그를 지정합니다.
- KMS 키를 삭제합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 시나리오를 실행합니다.

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.kms.model.AlreadyExistsException;
import software.amazon.awssdk.services.kms.model.DisabledException;
import software.amazon.awssdk.services.kms.model.EnableKeyRotationResponse;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.model.NotFoundException;
import software.amazon.awssdk.services.kms.model.RevokeGrantResponse;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * 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 String accountId = "";

    private static final Logger logger = LoggerFactory.getLogger(KMSScenario.class);

    static KMSActions kmsActions = new KMSActions();

    static Scanner scanner = new Scanner(System.in);

    static String aliasName = "alias/dev-encryption-key";

    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) {
            logger.info(usage);
```

```
            return;
        }
        String granteePrincipal = args[0];
        String policyName = "default";

        accountId = kmsActions.getAccountId();
        String keyDesc = "Created by the AWS KMS API";

        logger.info(DASHES);
        logger.info("""
            Welcome to the AWS Key Management SDK Basics 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 Basics scenario creates two key types:

            - A symmetric encryption key is used to encrypt and decrypt data.
            - An asymmetric key used to digitally sign data.

            Let's get started...
            """);
        waitForInputToContinue(scanner);

        try {
        // Run the methods that belong to this scenario.
        String targetKeyId = runScenario(granteePrincipal, keyDesc, policyName);
        requestDeleteResources(aliasName, targetKeyId);

        } catch (Throwable rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
```

```
        }
    }

    private static String runScenario(String granteePrincipal, String keyDesc,
String policyName) throws Throwable {
        logger.info(DASHES);
        logger.info("1. Create a symmetric KMS key\n");
        logger.info("First, the program will creates a symmetric KMS key that you
can used to encrypt and decrypt data.");
        waitForInputToContinue(scanner);
        String targetKeyId;
        try {
            CompletableFuture<String> futureKeyId =
kmsActions.createKeyAsync(keyDesc);
            targetKeyId = futureKeyId.join();
            logger.info("A symmetric key was successfully created " + targetKeyId);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("""
            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.
             """);
        waitForInputToContinue(scanner);
        boolean isEnabled;
        try {
            CompletableFuture<Boolean> futureIsKeyEnabled =
kmsActions.isKeyEnabledAsync(targetKeyId);
            isEnabled = futureIsKeyEnabled.join();
            logger.info("Is the key enabled? {}", isEnabled);
```

```
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }

        if (!isEnabled)
            try {
                CompletableFuture<Void> future =
kmsActions.enableKeyAsync(targetKeyId);
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: " + rt.getMessage());
                }
                throw cause;
            }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("3. Encrypt data using the symmetric KMS key");
        String plaintext = "Hello, AWS KMS!";
        logger.info("""
            One of the main uses of symmetric keys is to encrypt and decrypt data.
            Next, the code encrypts the string {} with the SYMMETRIC_DEFAULT
encryption algorithm.
            """, plaintext);
        waitForInputToContinue(scanner);
        SdkBytes encryptedData;
        try {
            CompletableFuture<SdkBytes> future =
kmsActions.encryptDataAsync(targetKeyId, plaintext);
            encryptedData = future.join();
```

```
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof DisabledException kmsDisabledEx) {
                logger.info("KMS error occurred due to a disabled
key: Error message: {}, Error code {}", kmsDisabledEx.getMessage(),
kmsDisabledEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("4. Create an alias");
        logger.info("""

            The alias name should be prefixed with 'alias/'.
            The default, 'alias/dev-encryption-key'.
             """);
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<Void> future =
kmsActions.createCustomAliasAsync(targetKeyId, aliasName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof AlreadyExistsException kmsExistsEx) {
                if (kmsExistsEx.getMessage().contains("already exists")) {
                    logger.info("The alias '" + aliasName + "' already exists.
Moving on...");
                }
            } else {
                logger.error("An unexpected error occurred: " + rt.getMessage(),
rt);
                deleteKey(targetKeyId);
                throw cause;
            }
        }
        waitForInputToContinue(scanner);
```

```
        logger.info(DASHES);
        logger.info("5. List all of your aliases");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Object> future = kmsActions.listAllAliasesAsync();
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("6. Enable automatic rotation of the KMS key");
        logger.info("""

            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);
        try {
            CompletableFuture<EnableKeyRotationResponse> future =
kmsActions.enableKeyRotationAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
```

```
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("""
            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 = null;
        try {
            CompletableFuture<String> futureGrantId =
kmsActions.grantKeyAsync(targetKeyId, granteePrincipal);
            grantId = futureGrantId.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. List grants for the KMS key");
        waitForInputToContinue(scanner);
        try {
```

```
            CompletableFuture<Object> future =
kmsActions.displayGrantIdsAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("9. Revoke the grant");
        logger.info("""
            The revocation of a grant immediately removes the permissions and access
that the grant had provided.
            This means that any principal (user, role, or service) that was granted
access to perform specific
            KMS operations on a KMS key will no longer be able to perform those
operations.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<RevokeGrantResponse> future =
kmsActions.revokeKeyGrantAsync(targetKeyId, grantId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                if (kmsEx.getMessage().contains("Grant does not exist")) {
                    logger.info("The grant ID '" + grantId + "' does not exist.
Moving on...");
                } else {
                    logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                    throw cause;
```

```
                }
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
                deleteAliasName(aliasName);
                deleteKey(targetKeyId);
                throw cause;
            }
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("10. Decrypt the data\n");
        logger.info("""
            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 decryptedData = "";
        try {
            CompletableFuture<String> future =
kmsActions.decryptDataAsync(encryptedData, targetKeyId);
            decryptedData = future.join();
            logger.info("Decrypted data: " + decryptedData);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        logger.info("Decrypted text is: " + decryptedData);
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("11. Replace a key policy\n");
        logger.info("""
```

```
            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);
        try {
            CompletableFuture<Boolean> future =
kmsActions.replacePolicyAsync(targetKeyId, policyName, accountId);
            boolean success = future.join();
            if (success) {
                logger.info("Key policy replacement succeeded.");
            } else {
                logger.error("Key policy replacement failed.");
            }

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
```

```
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("12. Get the key policy\n");
        logger.info("The next bit of code that runs gets the key policy to make sure
it exists.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
kmsActions.getKeyPolicyAsync(targetKeyId, policyName);
            String policy = future.join();
            if (!policy.isEmpty()) {
                logger.info("Retrieved policy: " + policy);
            }

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("13. Create an asymmetric KMS key and sign your data\n");
        logger.info("""
             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);
        try {
            CompletableFuture<Boolean> future = kmsActions.signVerifyDataAsync();
            boolean success = future.join();
            if (success) {
                logger.info("Sign and verify data operation succeeded.");
            } else {
                logger.error("Sign and verify data operation failed.");
            }

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("14. Tag your symmetric KMS Key\n");
        logger.info("""
            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);
        try {
            CompletableFuture<Void> future = kmsActions.tagKMSKeyAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
```

```
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);
        return targetKeyId;
    }

    // Deletes KMS resources with user input.
    private static void requestDeleteResources(String aliasName, String targetKeyId)
{
        logger.info(DASHES);
        logger.info("15. Schedule the deletion of the KMS key\n");
        logger.info("""
            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.
            """);
        logger.info("Would you like to delete the Key Management resources? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete the AWS KMS resources.");
            waitForInputToContinue(scanner);
            try {
                CompletableFuture<Void> future =
kmsActions.deleteSpecificAliasAsync(aliasName);
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
```

```
                    logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: " + rt.getMessage());
                }
            }
            waitForInputToContinue(scanner);
            try {
                CompletableFuture<Void> future =
kmsActions.disableKeyAsync(targetKeyId);
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: " + rt.getMessage());
                }
            }

            try {
                CompletableFuture<Void> future =
kmsActions.deleteKeyAsync(targetKeyId);
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: " + rt.getMessage());
                }
            }

        } else {
            logger.info("The Key Management resources will not be deleted");
        }

        logger.info(DASHES);
        logger.info("This concludes the AWS Key Management SDK scenario");
        logger.info(DASHES);
```

```
    }

    // This method is invoked from Exceptions to clean up the resources.
    private static void deleteKey(String targetKeyId) {
        try {
            CompletableFuture<Void> future =
kmsActions.disableKeyAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }

        try {
            CompletableFuture<Void> future = kmsActions.deleteKeyAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }

    // This method is invoked from Exceptions to clean up the resources.
    private static void deleteAliasName(String aliasName) {
        try {
            CompletableFuture<Void> future =
kmsActions.deleteSpecificAliasAsync(aliasName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
```

```
                logger.info("KMS error occurred: Error message: {}, Error code {}",
 kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

KMS 작업을 래핑하는 클래스를 정의합니다.

```
public class KMSActions {
    private static final Logger logger = LoggerFactory.getLogger(KMSActions.class);
    private static KmsAsyncClient kmsAsyncClient;

    /**
     * Retrieves an asynchronous AWS Key Management Service (KMS) client.
     * <p>
     * This method creates and returns a singleton instance of the KMS async client,
 with the following configurations:
     * <ul>
     *   <li>Max concurrency: 100</li>
     *   <li>Connection timeout: 60 seconds</li>
     *   <li>Read timeout: 60 seconds</li>
     *   <li>Write timeout: 60 seconds</li>
```

```
     *   <li>API call timeout: 2 minutes</li>
     *   <li>API call attempt timeout: 90 seconds</li>
     *   <li>Retry policy: up to 3 retries</li>
     *   <li>Credentials provider: environment variable credentials provider</li>
     * </ul>
     * <p>
     * If the client instance has already been created, it is returned instead of
creating a new one.
     *
     * @return the KMS async client instance
     */
    private static KmsAsyncClient getAsyncClient() {
        if (kmsAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryPolicy(RetryPolicy.builder()
                    .numRetries(3)
                    .build())
                .build();

            kmsAsyncClient = KmsAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return kmsAsyncClient;
    }

    /**
     * Creates a new symmetric encryption key asynchronously.
     *
     * @param keyDesc the description of the key to be created
     * @return a {@link CompletableFuture} that completes with the ID of the newly
created key
     * @throws RuntimeException if an error occurs while creating the key
```

```
     */
    public CompletableFuture<String> createKeyAsync(String keyDesc) {
        CreateKeyRequest keyRequest = CreateKeyRequest.builder()
            .description(keyDesc)
            .keySpec(KeySpec.SYMMETRIC_DEFAULT)
            .keyUsage(KeyUsageType.ENCRYPT_DECRYPT)
            .build();

        return getAsyncClient().createKey(keyRequest)
            .thenApply(resp -> resp.keyMetadata().keyId())
            .exceptionally(ex -> {
                throw new RuntimeException("An error occurred while creating the
key: " + ex.getMessage(), ex);
            });
    }

    /**
     * Asynchronously checks if a specified key is enabled.
     *
     * @param keyId the ID of the key to check
     * @return a {@link CompletableFuture} that, when completed, indicates whether
the key is enabled or not
     *
     * @throws RuntimeException if an exception occurs while checking the key state
     */
    public CompletableFuture<Boolean> isKeyEnabledAsync(String keyId) {
        DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
            .keyId(keyId)
            .build();

        CompletableFuture<DescribeKeyResponse> responseFuture =
getAsyncClient().describeKey(keyRequest);
        return responseFuture.whenComplete((resp, ex) -> {
            if (resp != null) {
                KeyState keyState = resp.keyMetadata().keyState();
                if (keyState == KeyState.ENABLED) {
                    logger.info("The key is enabled.");
                } else {
                    logger.info("The key is not enabled. Key state: {}", keyState);
                }
            } else {
                throw new RuntimeException(ex);
            }
        }).thenApply(resp -> resp.keyMetadata().keyState() == KeyState.ENABLED);
```

```
    }

    /**
     * Asynchronously enables the specified key.
     *
     * @param keyId the ID of the key to enable
     * @return a {@link CompletableFuture} that completes when the key has been
enabled
     */
    public CompletableFuture<Void> enableKeyAsync(String keyId) {
        EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
            .keyId(keyId)
            .build();

        CompletableFuture<EnableKeyResponse> responseFuture =
getAsyncClient().enableKey(enableKeyRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("Key with ID [{}] has been enabled.", keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while enabling
key: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred while
enabling key: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(response -> null);
    }

    /**
     * Encrypts the given text asynchronously using the specified KMS client and key
ID.
     *
     * @param keyId the ID of the KMS key to use for encryption
     * @param text the text to encrypt
     * @return a CompletableFuture that completes with the encrypted data as an
SdkBytes object
     */
    public CompletableFuture<SdkBytes> encryptDataAsync(String keyId, String text) {
        SdkBytes myBytes = SdkBytes.fromUtf8String(text);
```

```
        EncryptRequest encryptRequest = EncryptRequest.builder()
            .keyId(keyId)
            .plaintext(myBytes)
            .build();

        CompletableFuture<EncryptResponse> responseFuture =
getAsyncClient().encrypt(encryptRequest).toCompletableFuture();
        return responseFuture.whenComplete((response, ex) -> {
            if (response != null) {
                String algorithm = response.encryptionAlgorithm().toString();
                logger.info("The string was encrypted with algorithm {}.",
algorithm);
            } else {
                throw new RuntimeException(ex);
            }
        }).thenApply(EncryptResponse::ciphertextBlob);
    }

    /**
     * Creates a custom alias for the specified target key asynchronously.
     *
     * @param targetKeyId the ID of the target key for the alias
     * @param aliasName   the name of the alias to create
     * @return a {@link CompletableFuture} that completes when the alias creation
operation is finished
     */
    public CompletableFuture<Void> createCustomAliasAsync(String targetKeyId, String
aliasName) {
        CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
            .aliasName(aliasName)
            .targetKeyId(targetKeyId)
            .build();

        CompletableFuture<CreateAliasResponse> responseFuture =
getAsyncClient().createAlias(aliasRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("{} was successfully created.", aliasName);
            } else {
                if (exception instanceof ResourceExistsException) {
                    logger.info("Alias [{}] already exists. Moving on...",
aliasName);
                } else if (exception instanceof KmsException kmsEx) {
```

```
                    throw new RuntimeException("KMS error occurred while creating
alias: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred while
creating alias: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(response -> null);
    }

    /**
     * Asynchronously lists all the aliases in the current AWS account.
     *
     * @return a {@link CompletableFuture} that completes when the list of aliases
has been processed
     */
    public CompletableFuture<Object> listAllAliasesAsync() {
        ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
            .limit(15)
            .build();

        ListAliasesPublisher paginator =
getAsyncClient().listAliasesPaginator(aliasesRequest);
        return paginator.subscribe(response -> {
                response.aliases().forEach(alias ->
                    logger.info("The alias name is: " + alias.aliasName())
                );
            })
            .thenApply(v -> null)
            .exceptionally(ex -> {
                if (ex.getCause() instanceof KmsException) {
                    KmsException e = (KmsException) ex.getCause();
                    throw new RuntimeException("A KMS exception occurred: " +
e.getMessage());
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage());
                }
            });
    }

    /**
```

```
    * Enables key rotation asynchronously for the specified key ID.
    *
    * @param keyId the ID of the key for which to enable key rotation
    * @return a CompletableFuture that represents the asynchronous operation of
enabling key rotation
    * @throws RuntimeException if there was an error enabling key rotation, either
due to a KMS exception or an unexpected error
    */
   public CompletableFuture<EnableKeyRotationResponse>
enableKeyRotationAsync(String keyId) {
       EnableKeyRotationRequest enableKeyRotationRequest =
EnableKeyRotationRequest.builder()
           .keyId(keyId)
           .build();

       CompletableFuture<EnableKeyRotationResponse> responseFuture =
getAsyncClient().enableKeyRotation(enableKeyRotationRequest);
       responseFuture.whenComplete((response, exception) -> {
           if (exception == null) {
               logger.info("Key rotation has been enabled for key with id [{}]",
keyId);
           } else {
               if (exception instanceof KmsException kmsEx) {
                   throw new RuntimeException("Failed to enable key rotation: " +
kmsEx.getMessage(), kmsEx);
               } else {
                   throw new RuntimeException("An unexpected error occurred: " +
exception.getMessage(), exception);
               }
           }
       });

       return responseFuture;
   }

   /**
    * Grants permissions to a specified principal on a customer master key (CMK)
asynchronously.
    *
    * @param keyId             The unique identifier for the customer master key
(CMK) that the grant applies to.
    * @param granteePrincipal  The principal that is given permission to perform
the operations that the grant permits on the CMK.
```

```
    * @return A {@link CompletableFuture} that, when completed, contains the ID of
the created grant.
    * @throws RuntimeException If an error occurs during the grant creation
process.
    */
   public CompletableFuture<String> grantKeyAsync(String keyId, String
granteePrincipal) {
       List<GrantOperation> grantPermissions = List.of(
           GrantOperation.ENCRYPT,
           GrantOperation.DECRYPT,
           GrantOperation.DESCRIBE_KEY
       );

       CreateGrantRequest grantRequest = CreateGrantRequest.builder()
           .keyId(keyId)
           .name("grant1")
           .granteePrincipal(granteePrincipal)
           .operations(grantPermissions)
           .build();

       CompletableFuture<CreateGrantResponse> responseFuture =
getAsyncClient().createGrant(grantRequest);
       responseFuture.whenComplete((response, ex) -> {
           if (ex == null) {
               logger.info("Grant created successfully with ID: " +
response.grantId());
           } else {
               if (ex instanceof KmsException kmsEx) {
                   throw new RuntimeException("Failed to create grant: " +
kmsEx.getMessage(), kmsEx);
               } else {
                   throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage(), ex);
               }
           }
       });

       return responseFuture.thenApply(CreateGrantResponse::grantId);
   }

   /**
    * Asynchronously displays the grant IDs for the specified key ID.
    *
    * @param keyId the ID of the AWS KMS key for which to list the grants
```

```
    * @return a {@link CompletableFuture} that, when completed, will be null if
the operation succeeded, or will throw a {@link RuntimeException} if the operation
failed
    * @throws RuntimeException if there was an error listing the grants, either due
to an {@link KmsException} or an unexpected error
    */
   public CompletableFuture<Object> displayGrantIdsAsync(String keyId) {
       ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
           .keyId(keyId)
           .limit(15)
           .build();

       ListGrantsPublisher paginator =
getAsyncClient().listGrantsPaginator(grantsRequest);
       return paginator.subscribe(response -> {
               response.grants().forEach(grant -> {
                   logger.info("The grant Id is: " + grant.grantId());
               });
           })
           .thenApply(v -> null)
           .exceptionally(ex -> {
               Throwable cause = ex.getCause();
               if (cause instanceof KmsException) {
                   throw new RuntimeException("Failed to list grants: " +
cause.getMessage(), cause);
               } else {
                   throw new RuntimeException("An unexpected error occurred: " +
cause.getMessage(), cause);
               }
           });
   }

   /**
    * Revokes a grant for the specified AWS KMS key asynchronously.
    *
    * @param keyId   The ID or key ARN of the AWS KMS key.
    * @param grantId The identifier of the grant to be revoked.
    * @return A {@link CompletableFuture} representing the asynchronous operation
of revoking the grant.
    *         The {@link CompletableFuture} will complete with a {@link
RevokeGrantResponse} object
    *         if the operation is successful, or with a {@code null} value if an
error occurs.
    */
```

```
   public CompletableFuture<RevokeGrantResponse> revokeKeyGrantAsync(String keyId,
String grantId) {
       RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
           .keyId(keyId)
           .grantId(grantId)
           .build();

       CompletableFuture<RevokeGrantResponse> responseFuture =
getAsyncClient().revokeGrant(grantRequest);
       responseFuture.whenComplete((response, exception) -> {
           if (exception == null) {
               logger.info("Grant ID: [" + grantId + "] was successfully
revoked!");
           } else {
               if (exception instanceof KmsException kmsEx) {
                   if (kmsEx.getMessage().contains("Grant does not exist")) {
                       logger.info("The grant ID '" + grantId + "' does not exist.
Moving on...");
                   } else {
                       throw new RuntimeException("KMS error occurred: " +
kmsEx.getMessage(), kmsEx);
                   }
               } else {
                   throw new RuntimeException("An unexpected error occurred: " +
exception.getMessage(), exception);
               }
           }
       });

       return responseFuture;
   }


   /**
    * Asynchronously decrypts the given encrypted data using the specified key ID.
    *
    * @param encryptedData The encrypted data to be decrypted.
    * @param keyId The ID of the key to be used for decryption.
    * @return A CompletableFuture that, when completed, will contain the decrypted
data as a String.
    *         If an error occurs during the decryption process, the
CompletableFuture will complete
    *         exceptionally with the error, and the method will return an empty
String.
```

```
     */
    public CompletableFuture<String> decryptDataAsync(SdkBytes encryptedData, String
keyId) {
        DecryptRequest decryptRequest = DecryptRequest.builder()
            .ciphertextBlob(encryptedData)
            .keyId(keyId)
            .build();

        CompletableFuture<DecryptResponse> responseFuture =
getAsyncClient().decrypt(decryptRequest);
        responseFuture.whenComplete((decryptResponse, exception) -> {
            if (exception == null) {
                logger.info("Data decrypted successfully for key ID: " + keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while decrypting
data: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred while
decrypting data: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(decryptResponse ->
decryptResponse.plaintext().asString(StandardCharsets.UTF_8));
    }

    /**
     * Asynchronously replaces the policy for the specified KMS key.
     *
     * @param keyId       the ID of the KMS key to replace the policy for
     * @param policyName  the name of the policy to be replaced
     * @param accountId   the AWS account ID to be used in the policy
     * @return a {@link CompletableFuture} that completes with a boolean indicating
     *         whether the policy replacement was successful or not
     */
    public CompletableFuture<Boolean> replacePolicyAsync(String keyId, String
policyName, String accountId) {
        String policy = """
    {
      "Version":"2012-10-17",
      "Statement": [{
        "Effect": "Allow",
```

```
        "Principal": {"AWS": "arn:aws:iam::%s:root"},
        "Action": "kms:*",
        "Resource": "*"
      }]
   }
   """.formatted(accountId);

       PutKeyPolicyRequest keyPolicyRequest = PutKeyPolicyRequest.builder()
           .keyId(keyId)
           .policyName(policyName)
           .policy(policy)
           .build();

       // First, get the current policy to check if it exists
       return getAsyncClient().getKeyPolicy(r ->
r.keyId(keyId).policyName(policyName))
           .thenCompose(response -> {
               logger.info("Current policy exists. Replacing it...");
               return getAsyncClient().putKeyPolicy(keyPolicyRequest);
           })
           .thenApply(putPolicyResponse -> {
               logger.info("The key policy has been replaced.");
               return true;
           })
           .exceptionally(throwable -> {
               if (throwable.getCause() instanceof LimitExceededException) {
                   logger.error("Cannot replace policy, as only one policy is
allowed per key.");
                   return false;
               }
               throw new RuntimeException("Error replacing policy", throwable);
           });
   }


   /**
    * Asynchronously retrieves the key policy for the specified key ID and policy
name.
    *
    * @param keyId       the ID of the AWS KMS key for which to retrieve the policy
    * @param policyName the name of the key policy to retrieve
    * @return a {@link CompletableFuture} that, when completed, contains the key
policy as a {@link String}
    */
```

```
    public CompletableFuture<String> getKeyPolicyAsync(String keyId, String
 policyName) {
        GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
            .keyId(keyId)
            .policyName(policyName)
            .build();

        return getAsyncClient().getKeyPolicy(policyRequest)
            .thenApply(response -> {
                String policy = response.policy();
                logger.info("The response is: " + policy);
                return policy;
            })
            .exceptionally(ex -> {
                throw new RuntimeException("Failed to get key policy", ex);
            });
    }

    /**
     * Asynchronously signs and verifies data using AWS KMS.
     *
     * <p>The method performs the following steps:
     * <ol>
     *     <li>Creates an AWS KMS key with the specified key spec, key usage, and
 origin.</li>
     *     <li>Signs the provided message using the created KMS key and the RSASSA-
PSS-SHA-256 algorithm.</li>
     *     <li>Verifies the signature of the message using the created KMS key and
 the RSASSA-PSS-SHA-256 algorithm.</li>
     * </ol>
     *
     * @return a {@link CompletableFuture} that completes with the result of the
 signature verification,
     *         {@code true} if the signature is valid, {@code false} otherwise.
     * @throws KmsException if any error occurs during the KMS operations.
     * @throws RuntimeException if an unexpected error occurs.
     */
    public CompletableFuture<Boolean> signVerifyDataAsync() {
        String signMessage = "Here is the message that will be digitally signed";

        // Create an AWS KMS key used to digitally sign data.
        CreateKeyRequest createKeyRequest = CreateKeyRequest.builder()
            .keySpec(KeySpec.RSA_2048)
            .keyUsage(KeyUsageType.SIGN_VERIFY)
```

```
            .origin(OriginType.AWS_KMS)
            .build();

        return getAsyncClient().createKey(createKeyRequest)
            .thenCompose(createKeyResponse -> {
                String keyId = createKeyResponse.keyMetadata().keyId();

                SdkBytes messageBytes = SdkBytes.fromString(signMessage,
Charset.defaultCharset());
                SignRequest signRequest = SignRequest.builder()
                    .keyId(keyId)
                    .message(messageBytes)
                    .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                    .build();

                return getAsyncClient().sign(signRequest)
                    .thenCompose(signResponse -> {
                        byte[] signedBytes = signResponse.signature().asByteArray();

                        VerifyRequest verifyRequest = VerifyRequest.builder()
                            .keyId(keyId)

.message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset())))

.signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))

.signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                            .build();

                        return getAsyncClient().verify(verifyRequest)
                            .thenApply(verifyResponse -> {
                                return (boolean) verifyResponse.signatureValid();
                            });
                    });
            })
            .exceptionally(throwable -> {
               throw new RuntimeException("Failed to sign or verify data",
throwable);
            });
    }

    /**
     * Asynchronously tags a KMS key with a specific tag.
     *
```

```
     * @param keyId the ID of the KMS key to be tagged
     * @return a {@link CompletableFuture} that completes when the tagging operation
is finished
     */
    public CompletableFuture<Void> tagKMSKeyAsync(String keyId) {
        Tag tag = Tag.builder()
            .tagKey("Environment")
            .tagValue("Production")
            .build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .keyId(keyId)
            .tags(tag)
            .build();

        return getAsyncClient().tagResource(tagResourceRequest)
            .thenRun(() -> {
                logger.info("{} key was tagged", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to tag the KMS key", throwable);
            });
    }

    /**
     * Deletes a specific KMS alias asynchronously.
     *
     * @param aliasName the name of the alias to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous operation
of deleting the specified alias
     */
    public CompletableFuture<Void> deleteSpecificAliasAsync(String aliasName) {
        DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
            .aliasName(aliasName)
            .build();

        return getAsyncClient().deleteAlias(deleteAliasRequest)
            .thenRun(() -> {
                logger.info("Alias {} has been deleted successfully", aliasName);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to delete alias: " + aliasName,
throwable);
            });
```

```
    }

    /**
     * Asynchronously disables the specified AWS Key Management Service (KMS) key.
     *
     * @param keyId the ID or Amazon Resource Name (ARN) of the KMS key to be
disabled
     * @return a CompletableFuture that, when completed, indicates that the key has
been disabled successfully
     */
    public CompletableFuture<Void> disableKeyAsync(String keyId) {
        DisableKeyRequest keyRequest = DisableKeyRequest.builder()
            .keyId(keyId)
            .build();

        return getAsyncClient().disableKey(keyRequest)
            .thenRun(() -> {
                logger.info("Key {} has been disabled successfully",keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to disable key: " + keyId,
throwable);
            });
    }

    /**
     * Deletes a KMS key asynchronously.
     *
     * <p><strong>Warning:</strong> 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
becomes unrecoverable.
     * This means that any files, databases, or other data that were encrypted using
the deleted KMS key
     * will become permanently inaccessible. Exercise extreme caution when deleting
KMS keys.</p>
     *
     * @param keyId the ID of the KMS key to delete
     * @return a {@link CompletableFuture} that completes when the key deletion is
scheduled
     */
    public CompletableFuture<Void> deleteKeyAsync(String keyId) {
        ScheduleKeyDeletionRequest deletionRequest =
ScheduleKeyDeletionRequest.builder()
```

```
            .keyId(keyId)
            .pendingWindowInDays(7)
            .build();

        return getAsyncClient().scheduleKeyDeletion(deletionRequest)
            .thenRun(() -> {
                logger.info("Key {} will be deleted in 7 days", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to schedule key deletion for key
 ID: " + keyId, throwable);
            });
    }


    public String getAccountId(){
        try (StsClient stsClient = StsClient.create()){
            GetCallerIdentityResponse callerIdentity =
 stsClient.getCallerIdentity();
            return callerIdentity.account();
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - CreateAlias
  - CreateGrant
  - CreateKey
  - Decrypt
  - DescribeKey
  - DisableKey
  - EnableKey
  - 암호화
  - GetKeyPolicy
  - ListAliases
  - ListGrants
  - ListKeys

- RevokeGrant
- ScheduleKeyDeletion
- Sign
- TagResource

작업

`CreateAlias`

다음 코드 예시는 `CreateAlias`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Creates a custom alias for the specified target key asynchronously.
    *
    * @param targetKeyId the ID of the target key for the alias
    * @param aliasName   the name of the alias to create
    * @return a {@link CompletableFuture} that completes when the alias creation
operation is finished
    */
   public CompletableFuture<Void> createCustomAliasAsync(String targetKeyId, String
aliasName) {
       CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
           .aliasName(aliasName)
           .targetKeyId(targetKeyId)
           .build();

       CompletableFuture<CreateAliasResponse> responseFuture =
getAsyncClient().createAlias(aliasRequest);
       responseFuture.whenComplete((response, exception) -> {
           if (exception == null) {
               logger.info("{} was successfully created.", aliasName);
           } else {
```

```
                if (exception instanceof ResourceExistsException) {
                    logger.info("Alias [{}] already exists. Moving on...",
aliasName);
                } else if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while creating
alias: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred while
creating alias: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(response -> null);
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateAlias를 참조하세요.

`CreateGrant`

다음 코드 예시는 `CreateGrant`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Grants permissions to a specified principal on a customer master key (CMK)
asynchronously.
     *
     * @param keyId             The unique identifier for the customer master key
(CMK) that the grant applies to.
     * @param granteePrincipal  The principal that is given permission to perform
the operations that the grant permits on the CMK.
     * @return A {@link CompletableFuture} that, when completed, contains the ID of
the created grant.
```

```
     * @throws RuntimeException If an error occurs during the grant creation
process.
     */
    public CompletableFuture<String> grantKeyAsync(String keyId, String
granteePrincipal) {
        List<GrantOperation> grantPermissions = List.of(
            GrantOperation.ENCRYPT,
            GrantOperation.DECRYPT,
            GrantOperation.DESCRIBE_KEY
        );

        CreateGrantRequest grantRequest = CreateGrantRequest.builder()
            .keyId(keyId)
            .name("grant1")
            .granteePrincipal(granteePrincipal)
            .operations(grantPermissions)
            .build();

        CompletableFuture<CreateGrantResponse> responseFuture =
getAsyncClient().createGrant(grantRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex == null) {
                logger.info("Grant created successfully with ID: " +
response.grantId());
            } else {
                if (ex instanceof KmsException kmsEx) {
                    throw new RuntimeException("Failed to create grant: " +
kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage(), ex);
                }
            }
        });

        return responseFuture.thenApply(CreateGrantResponse::grantId);
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateGrant를 참조하세요.

`CreateKey`

다음 코드 예시는 `CreateKey`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Creates a new symmetric encryption key asynchronously.
     *
     * @param keyDesc the description of the key to be created
     * @return a {@link CompletableFuture} that completes with the ID of the newly
created key
     * @throws RuntimeException if an error occurs while creating the key
     */
    public CompletableFuture<String> createKeyAsync(String keyDesc) {
        CreateKeyRequest keyRequest = CreateKeyRequest.builder()
            .description(keyDesc)
            .keySpec(KeySpec.SYMMETRIC_DEFAULT)
            .keyUsage(KeyUsageType.ENCRYPT_DECRYPT)
            .build();

        return getAsyncClient().createKey(keyRequest)
            .thenApply(resp -> resp.keyMetadata().keyId())
            .exceptionally(ex -> {
                throw new RuntimeException("An error occurred while creating the
key: " + ex.getMessage(), ex);
            });
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateKey를 참조하세요.

`Decrypt`

다음 코드 예시는 `Decrypt`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Asynchronously decrypts the given encrypted data using the specified key ID.
     *
     * @param encryptedData The encrypted data to be decrypted.
     * @param keyId The ID of the key to be used for decryption.
     * @return A CompletableFuture that, when completed, will contain the decrypted
data as a String.
     *         If an error occurs during the decryption process, the
CompletableFuture will complete
     *         exceptionally with the error, and the method will return an empty
String.
     */
    public CompletableFuture<String> decryptDataAsync(SdkBytes encryptedData, String
keyId) {
        DecryptRequest decryptRequest = DecryptRequest.builder()
            .ciphertextBlob(encryptedData)
            .keyId(keyId)
            .build();

        CompletableFuture<DecryptResponse> responseFuture =
getAsyncClient().decrypt(decryptRequest);
        responseFuture.whenComplete((decryptResponse, exception) -> {
            if (exception == null) {
                logger.info("Data decrypted successfully for key ID: " + keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while decrypting
data: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred while
decrypting data: " + exception.getMessage(), exception);
                }
            }
        });
```

```
        return responseFuture.thenApply(decryptResponse ->
decryptResponse.plaintext().asString(StandardCharsets.UTF_8));
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 Decrypt를 참조하세요.

`DeleteAlias`

다음 코드 예시는 `DeleteAlias`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Deletes a specific KMS alias asynchronously.
     *
     * @param aliasName the name of the alias to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous operation
of deleting the specified alias
     */
    public CompletableFuture<Void> deleteSpecificAliasAsync(String aliasName) {
        DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
            .aliasName(aliasName)
            .build();

        return getAsyncClient().deleteAlias(deleteAliasRequest)
            .thenRun(() -> {
                logger.info("Alias {} has been deleted successfully", aliasName);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to delete alias: " + aliasName,
throwable);
            });
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteAlias를 참조하세요.

`DescribeKey`

다음 코드 예시는 `DescribeKey`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Asynchronously checks if a specified key is enabled.
    *
    * @param keyId the ID of the key to check
    * @return a {@link CompletableFuture} that, when completed, indicates whether
the key is enabled or not
    *
    * @throws RuntimeException if an exception occurs while checking the key state
    */
   public CompletableFuture<Boolean> isKeyEnabledAsync(String keyId) {
       DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
           .keyId(keyId)
           .build();

       CompletableFuture<DescribeKeyResponse> responseFuture =
getAsyncClient().describeKey(keyRequest);
       return responseFuture.whenComplete((resp, ex) -> {
           if (resp != null) {
               KeyState keyState = resp.keyMetadata().keyState();
               if (keyState == KeyState.ENABLED) {
                   logger.info("The key is enabled.");
               } else {
                   logger.info("The key is not enabled. Key state: {}", keyState);
               }
           } else {
```

```
                throw new RuntimeException(ex);
            }
        }).thenApply(resp -> resp.keyMetadata().keyState() == KeyState.ENABLED);
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeKey를 참조하세요.

`DisableKey`

다음 코드 예시는 `DisableKey`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Asynchronously disables the specified AWS Key Management Service (KMS) key.
     *
     * @param keyId the ID or Amazon Resource Name (ARN) of the KMS key to be
disabled
     * @return a CompletableFuture that, when completed, indicates that the key has
been disabled successfully
     */
    public CompletableFuture<Void> disableKeyAsync(String keyId) {
        DisableKeyRequest keyRequest = DisableKeyRequest.builder()
            .keyId(keyId)
            .build();

        return getAsyncClient().disableKey(keyRequest)
            .thenRun(() -> {
                logger.info("Key {} has been disabled successfully",keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to disable key: " + keyId,
throwable);
            });
```

```
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DisableKey를 참조하세요.

`EnableKey`

다음 코드 예시는 `EnableKey`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Asynchronously enables the specified key.
    *
    * @param keyId the ID of the key to enable
    * @return a {@link CompletableFuture} that completes when the key has been
enabled
    */
   public CompletableFuture<Void> enableKeyAsync(String keyId) {
       EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
           .keyId(keyId)
           .build();

       CompletableFuture<EnableKeyResponse> responseFuture =
getAsyncClient().enableKey(enableKeyRequest);
       responseFuture.whenComplete((response, exception) -> {
           if (exception == null) {
               logger.info("Key with ID [{}] has been enabled.", keyId);
           } else {
               if (exception instanceof KmsException kmsEx) {
                   throw new RuntimeException("KMS error occurred while enabling
key: " + kmsEx.getMessage(), kmsEx);
               } else {
                   throw new RuntimeException("An unexpected error occurred while
enabling key: " + exception.getMessage(), exception);
               }
```

```
            }
        });

        return responseFuture.thenApply(response -> null);
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 EnableKey를 참조하세요.

`Encrypt`

다음 코드 예시는 `Encrypt`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Encrypts the given text asynchronously using the specified KMS client and key
ID.
     *
     * @param keyId the ID of the KMS key to use for encryption
     * @param text the text to encrypt
     * @return a CompletableFuture that completes with the encrypted data as an
SdkBytes object
     */
    public CompletableFuture<SdkBytes> encryptDataAsync(String keyId, String text) {
        SdkBytes myBytes = SdkBytes.fromUtf8String(text);
        EncryptRequest encryptRequest = EncryptRequest.builder()
            .keyId(keyId)
            .plaintext(myBytes)
            .build();

        CompletableFuture<EncryptResponse> responseFuture =
getAsyncClient().encrypt(encryptRequest).toCompletableFuture();
        return responseFuture.whenComplete((response, ex) -> {
            if (response != null) {
                String algorithm = response.encryptionAlgorithm().toString();
```

```
                logger.info("The string was encrypted with algorithm {}.",
algorithm);
            } else {
                throw new RuntimeException(ex);
            }
        }).thenApply(EncryptResponse::ciphertextBlob);
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 Encrypt를 참조하세요.

`ListAliases`

다음 코드 예시는 `ListAliases`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Asynchronously lists all the aliases in the current AWS account.
     *
     * @return a {@link CompletableFuture} that completes when the list of aliases
has been processed
     */
    public CompletableFuture<Object> listAllAliasesAsync() {
        ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
            .limit(15)
            .build();

        ListAliasesPublisher paginator =
getAsyncClient().listAliasesPaginator(aliasesRequest);
        return paginator.subscribe(response -> {
                response.aliases().forEach(alias ->
                    logger.info("The alias name is: " + alias.aliasName())
                );
            })
            .thenApply(v -> null)
```

```
            .exceptionally(ex -> {
                if (ex.getCause() instanceof KmsException) {
                    KmsException e = (KmsException) ex.getCause();
                    throw new RuntimeException("A KMS exception occurred: " +
e.getMessage());
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage());
                }
            });
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListAliases를 참조하세요.

`ListGrants`

다음 코드 예시는 `ListGrants`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Asynchronously displays the grant IDs for the specified key ID.
     *
     * @param keyId the ID of the AWS KMS key for which to list the grants
     * @return a {@link CompletableFuture} that, when completed, will be null if
the operation succeeded, or will throw a {@link RuntimeException} if the operation
failed
     * @throws RuntimeException if there was an error listing the grants, either due
to an {@link KmsException} or an unexpected error
     */
    public CompletableFuture<Object> displayGrantIdsAsync(String keyId) {
        ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
            .keyId(keyId)
            .limit(15)
            .build();
```

```
        ListGrantsPublisher paginator =
getAsyncClient().listGrantsPaginator(grantsRequest);
        return paginator.subscribe(response -> {
                response.grants().forEach(grant -> {
                    logger.info("The grant Id is: " + grant.grantId());
                });
            })
            .thenApply(v -> null)
            .exceptionally(ex -> {
                Throwable cause = ex.getCause();
                if (cause instanceof KmsException) {
                    throw new RuntimeException("Failed to list grants: " +
cause.getMessage(), cause);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
cause.getMessage(), cause);
                }
            });
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListGrants를 참조하세요.

`ListKeyPolicies`

다음 코드 예시는 `ListKeyPolicies`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Asynchronously retrieves the key policy for the specified key ID and policy
name.
     *
     * @param keyId       the ID of the AWS KMS key for which to retrieve the policy
     * @param policyName the name of the key policy to retrieve
```

```
     * @return a {@link CompletableFuture} that, when completed, contains the key
 policy as a {@link String}
     */
    public CompletableFuture<String> getKeyPolicyAsync(String keyId, String
 policyName) {
        GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
            .keyId(keyId)
            .policyName(policyName)
            .build();

        return getAsyncClient().getKeyPolicy(policyRequest)
            .thenApply(response -> {
                String policy = response.policy();
                logger.info("The response is: " + policy);
                return policy;
            })
            .exceptionally(ex -> {
                throw new RuntimeException("Failed to get key policy", ex);
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListKeyPolicies를 참조하세요.

`ListKeys`

다음 코드 예시는 `ListKeys`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.services.kms.KmsAsyncClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.paginators.ListKeysPublisher;
import java.util.concurrent.CompletableFuture;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * 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) {
        listAllKeys();
    }

    public static void listAllKeys() {
        KmsAsyncClient kmsAsyncClient = KmsAsyncClient.builder()
            .build();
        ListKeysRequest listKeysRequest = ListKeysRequest.builder()
            .limit(15)
            .build();

        /*
         * The `subscribe` method is required when using paginator methods in the
 AWS SDK
         * because paginator methods return an instance of a `ListKeysPublisher`,
 which is
         * based on a reactive stream. This allows asynchronous retrieval of
 paginated
         * results as they become available. By subscribing to the stream, we can
 process
         * each page of results as they are emitted.
         */
        ListKeysPublisher keysPublisher =
 kmsAsyncClient.listKeysPaginator(listKeysRequest);
        CompletableFuture<Void> future = keysPublisher
            .subscribe(r -> r.keys().forEach(key ->
                System.out.println("The key ARN is: " + key.keyArn() + ". The key Id
 is: " + key.keyId())))
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    System.err.println("Error occurred: " + exception.getMessage());
                } else {
                    System.out.println("Successfully listed all keys.");
                }
            });
```

```
        try {
            future.join();
        } catch (Exception e) {
            System.err.println("Failed to list keys: " + e.getMessage());
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListKeys를 참조하세요.

`RevokeGrant`

다음 코드 예시는 `RevokeGrant`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Revokes a grant for the specified AWS KMS key asynchronously.
    *
    * @param keyId   The ID or key ARN of the AWS KMS key.
    * @param grantId The identifier of the grant to be revoked.
    * @return A {@link CompletableFuture} representing the asynchronous operation
of revoking the grant.
    *         The {@link CompletableFuture} will complete with a {@link
RevokeGrantResponse} object
    *         if the operation is successful, or with a {@code null} value if an
error occurs.
    */
   public CompletableFuture<RevokeGrantResponse> revokeKeyGrantAsync(String keyId,
String grantId) {
       RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
           .keyId(keyId)
           .grantId(grantId)
           .build();
```

```
        CompletableFuture<RevokeGrantResponse> responseFuture =
getAsyncClient().revokeGrant(grantRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("Grant ID: [" + grantId + "] was successfully
revoked!");
            } else {
                if (exception instanceof KmsException kmsEx) {
                    if (kmsEx.getMessage().contains("Grant does not exist")) {
                        logger.info("The grant ID '" + grantId + "' does not exist.
Moving on...");
                    } else {
                        throw new RuntimeException("KMS error occurred: " +
kmsEx.getMessage(), kmsEx);
                    }
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
exception.getMessage(), exception);
                }
            }
        });

        return responseFuture;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 RevokeGrant를 참조하세요.

`ScheduleKeyDeletion`

다음 코드 예시는 `ScheduleKeyDeletion`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
```

```
    * Deletes a KMS key asynchronously.
    *
    * <p><strong>Warning:</strong> 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
becomes unrecoverable.
    * This means that any files, databases, or other data that were encrypted using
the deleted KMS key
    * will become permanently inaccessible. Exercise extreme caution when deleting
KMS keys.</p>
    *
    * @param keyId the ID of the KMS key to delete
    * @return a {@link CompletableFuture} that completes when the key deletion is
scheduled
    */
   public CompletableFuture<Void> deleteKeyAsync(String keyId) {
       ScheduleKeyDeletionRequest deletionRequest =
ScheduleKeyDeletionRequest.builder()
           .keyId(keyId)
           .pendingWindowInDays(7)
           .build();

       return getAsyncClient().scheduleKeyDeletion(deletionRequest)
           .thenRun(() -> {
               logger.info("Key {} will be deleted in 7 days", keyId);
           })
           .exceptionally(throwable -> {
               throw new RuntimeException("Failed to schedule key deletion for key
ID: " + keyId, throwable);
           });
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ScheduleKeyDeletion을 참조하세요.

Sign

다음 코드 예시는 Sign의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Asynchronously signs and verifies data using AWS KMS.
     *
     * <p>The method performs the following steps:
     * <ol>
     *     <li>Creates an AWS KMS key with the specified key spec, key usage, and
 origin.</li>
     *     <li>Signs the provided message using the created KMS key and the RSASSA-
PSS-SHA-256 algorithm.</li>
     *     <li>Verifies the signature of the message using the created KMS key and
 the RSASSA-PSS-SHA-256 algorithm.</li>
     * </ol>
     *
     * @return a {@link CompletableFuture} that completes with the result of the
 signature verification,
     *         {@code true} if the signature is valid, {@code false} otherwise.
     * @throws KmsException if any error occurs during the KMS operations.
     * @throws RuntimeException if an unexpected error occurs.
     */
    public CompletableFuture<Boolean> signVerifyDataAsync() {
        String signMessage = "Here is the message that will be digitally signed";

        // Create an AWS KMS key used to digitally sign data.
        CreateKeyRequest createKeyRequest = CreateKeyRequest.builder()
            .keySpec(KeySpec.RSA_2048)
            .keyUsage(KeyUsageType.SIGN_VERIFY)
            .origin(OriginType.AWS_KMS)
            .build();

        return getAsyncClient().createKey(createKeyRequest)
            .thenCompose(createKeyResponse -> {
                String keyId = createKeyResponse.keyMetadata().keyId();
```

```
                SdkBytes messageBytes = SdkBytes.fromString(signMessage,
Charset.defaultCharset());
                SignRequest signRequest = SignRequest.builder()
                    .keyId(keyId)
                    .message(messageBytes)
                    .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                    .build();

                return getAsyncClient().sign(signRequest)
                    .thenCompose(signResponse -> {
                        byte[] signedBytes = signResponse.signature().asByteArray();

                        VerifyRequest verifyRequest = VerifyRequest.builder()
                            .keyId(keyId)

.message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset())))

.signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))

.signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                            .build();

                        return getAsyncClient().verify(verifyRequest)
                            .thenApply(verifyResponse -> {
                                return (boolean) verifyResponse.signatureValid();
                            });
                    });
            })
            .exceptionally(throwable -> {
               throw new RuntimeException("Failed to sign or verify data",
throwable);
            });
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Sign을 참조하세요.

TagResource

다음 코드 예시는 TagResource의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Asynchronously tags a KMS key with a specific tag.
     *
     * @param keyId the ID of the KMS key to be tagged
     * @return a {@link CompletableFuture} that completes when the tagging operation
is finished
     */
    public CompletableFuture<Void> tagKMSKeyAsync(String keyId) {
        Tag tag = Tag.builder()
            .tagKey("Environment")
            .tagValue("Production")
            .build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .keyId(keyId)
            .tags(tag)
            .build();

        return getAsyncClient().tagResource(tagResourceRequest)
            .thenRun(() -> {
                logger.info("{} key was tagged", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to tag the KMS key", throwable);
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 TagResource를 참조하세요.

Java 2.x용 SDK를 사용하는 Lambda 예제

다음 코드 예제에서는 Lambda와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

AWS 커뮤니티 기여는 여러 팀이 생성하고 유지 관리하는 예입니다 AWS. 피드백을 제공하려면 연결된 리포지토리에 제공된 메커니즘을 사용합니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시작하기

Hello Lambda

다음 코드 예제에서는 Lambda 사용을 시작하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Lists the AWS Lambda functions associated with the current AWS account.
     *
     * @param awsLambda an instance of the {@link LambdaClient} class, which is used
to interact with the AWS Lambda service
     *
     * @throws LambdaException if an error occurs while interacting with the AWS
Lambda service
     */
    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);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListFunctions를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- IAM 역할과 Lambda 함수를 생성하고 핸들러 코드를 업로드합니다.

- 단일 파라미터로 함수를 간접적으로 간접 호출하고 결과를 가져옵니다.
- 함수 코드를 업데이트하고 환경 변수로 구성합니다.
- 새 파라미터로 함수를 간접적으로 간접 호출하고 결과를 가져옵니다. 반환된 실행 로그를 표시합니다.
- 계정의 함수를 나열합니다.

자세한 내용은 콘솔로 Lambda 함수 생성을 참조하세요.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/*
 *  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> <role> <handler> <bucketName> <key>\s

            Where:
                functionName - The name of the Lambda function.\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 != 5) {
             System.out.println(usage);
             return;
       }

       String functionName = args[0];
       String role = args[1];
       String handler = args[2];
       String bucketName = args[3];
       String key = args[4];
       LambdaClient awsLambda = LambdaClient.builder()
           .build();

       System.out.println(DASHES);
       System.out.println("Welcome to the AWS Lambda Basics scenario.");
       System.out.println(DASHES);

       System.out.println(DASHES);
       System.out.println("1. Create an AWS Lambda function.");
       String funArn = createLambdaFunction(awsLambda, functionName, key,
 bucketName, 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();
    }
```

```
    /**
     * Creates a new Lambda function in AWS using the AWS Lambda Java API.
     *
     * @param awsLambda    the AWS Lambda client used to interact with the AWS
Lambda service
     * @param functionName the name of the Lambda function to create
     * @param key          the S3 key of the function code
     * @param bucketName   the name of the S3 bucket containing the function code
     * @param role         the IAM role to assign to the Lambda function
     * @param handler      the fully qualified class name of the function handler
     * @return the Amazon Resource Name (ARN) of the created Lambda function
     */
    public static String createLambdaFunction(LambdaClient awsLambda,
                                               String functionName,
                                               String key,
                                               String bucketName,
                                               String role,
                                               String handler) {

        try {
            LambdaWaiter waiter = awsLambda.waiter();
            FunctionCode code = FunctionCode.builder()
                .s3Key(key)
                .s3Bucket(bucketName)
                .build();

            CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
                .functionName(functionName)
                .description("Created by the Lambda Java API")
                .code(code)
                .handler(handler)
                .runtime(Runtime.JAVA17)
                .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 e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    /**
     * Retrieves information about an AWS Lambda function.
     *
     * @param awsLambda    an instance of the {@link LambdaClient} class, which is
used to interact with the AWS Lambda service
     * @param functionName the name of the AWS Lambda function to retrieve
information about
     */
    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);
        }
    }

    /**
     * Lists the AWS Lambda functions associated with the current AWS account.
     *
     * @param awsLambda an instance of the {@link LambdaClient} class, which is used
to interact with the AWS Lambda service
     *
     * @throws LambdaException if an error occurs while interacting with the AWS
Lambda service
     */
    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);
        }
    }

    /**
     * Invokes a specific AWS Lambda function.
     *
     * @param awsLambda    an instance of {@link LambdaClient} to interact with the
AWS Lambda service
     * @param functionName the name of the AWS Lambda function to be invoked
     */
    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);
        }
    }

    /**
```

```
    * Updates the code for an AWS Lambda function.
    *
    * @param awsLambda  the AWS Lambda client
    * @param functionName the name of the Lambda function to update
    * @param bucketName the name of the S3 bucket where the function code is
located
    * @param key the key (file name) of the function code in the S3 bucket
    * @throws LambdaException if there is an error updating the function code
    */
   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);
       }
   }

   /**
    * Updates the configuration of an AWS Lambda function.
    *
```

```
    * @param awsLambda     the {@link LambdaClient} instance to use for the AWS
Lambda operation
    * @param functionName  the name of the AWS Lambda function to update
    * @param handler       the new handler for the AWS Lambda function
    *
    * @throws LambdaException if there is an error while updating the function
configuration
    */
   public static void updateFunctionConfiguration(LambdaClient awsLambda, String
functionName, String handler) {
       try {
           UpdateFunctionConfigurationRequest configurationRequest =
UpdateFunctionConfigurationRequest.builder()
                .functionName(functionName)
                .handler(handler)
                .runtime(Runtime.JAVA17)
                .build();

           awsLambda.updateFunctionConfiguration(configurationRequest);

       } catch (LambdaException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }

   /**
    * Deletes an AWS Lambda function.
    *
    * @param awsLambda     an instance of the {@link LambdaClient} class, which is
used to interact with the AWS Lambda service
    * @param functionName  the name of the Lambda function to be deleted
    *
    * @throws LambdaException if an error occurs while deleting the 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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 항목을 참조하세요.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - 간접 호출
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

작업

`CreateFunction`

다음 코드 예시는 `CreateFunction`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Creates a new Lambda function in AWS using the AWS Lambda Java API.
     *
     * @param awsLambda    the AWS Lambda client used to interact with the AWS
Lambda service
     * @param functionName the name of the Lambda function to create
```

```
     * @param key          the S3 key of the function code
     * @param bucketName   the name of the S3 bucket containing the function code
     * @param role         the IAM role to assign to the Lambda function
     * @param handler      the fully qualified class name of the function handler
     * @return the Amazon Resource Name (ARN) of the created Lambda function
     */
    public static String createLambdaFunction(LambdaClient awsLambda,
                                               String functionName,
                                               String key,
                                               String bucketName,
                                               String role,
                                               String handler) {

        try {
            LambdaWaiter waiter = awsLambda.waiter();
            FunctionCode code = FunctionCode.builder()
                .s3Key(key)
                .s3Bucket(bucketName)
                .build();

            CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
                .functionName(functionName)
                .description("Created by the Lambda Java API")
                .code(code)
                .handler(handler)
                .runtime(Runtime.JAVA17)
                .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 e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
```

```
        return "";
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateFunction을 참조하세요.

`DeleteFunction`

다음 코드 예시는 `DeleteFunction`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Deletes an AWS Lambda function.
     *
     * @param awsLambda     an instance of the {@link LambdaClient} class, which is
used to interact with the AWS Lambda service
     * @param functionName  the name of the Lambda function to be deleted
     *
     * @throws LambdaException if an error occurs while deleting the 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);
        }
```

```
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteFunction을 참조하세요.

`GetFunction`

다음 코드 예시는 `GetFunction`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Retrieves information about an AWS Lambda function.
    *
    * @param awsLambda    an instance of the {@link LambdaClient} class, which is
used to interact with the AWS Lambda service
    * @param functionName the name of the AWS Lambda function to retrieve
information about
    */
   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);
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetFunction을 참조하세요.

Invoke

다음 코드 예시는 Invoke의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Invokes a specific AWS Lambda function.
     *
     * @param awsLambda    an instance of {@link LambdaClient} to interact with the
AWS Lambda service
     * @param functionName the name of the AWS Lambda function to be invoked
     */
    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);
```

```
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 간접 호출을 참조하세요.

`UpdateFunctionCode`

다음 코드 예시는 `UpdateFunctionCode`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Updates the code for an AWS Lambda function.
     *
     * @param awsLambda  the AWS Lambda client
     * @param functionName the name of the Lambda function to update
     * @param bucketName the name of the S3 bucket where the function code is
located
     * @param key the key (file name) of the function code in the S3 bucket
     * @throws LambdaException if there is an error updating the function code
     */
    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);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateFunctionCode를 참조하세요.

`UpdateFunctionConfiguration`

다음 코드 예시는 `UpdateFunctionConfiguration`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Updates the configuration of an AWS Lambda function.
     *
     * @param awsLambda     the {@link LambdaClient} instance to use for the AWS
Lambda operation
     * @param functionName  the name of the AWS Lambda function to update
     * @param handler       the new handler for the AWS Lambda function
     *
     * @throws LambdaException if there is an error while updating the function
configuration
```

```
     */
    public static void updateFunctionConfiguration(LambdaClient awsLambda, String
functionName, String handler) {
        try {
            UpdateFunctionConfigurationRequest configurationRequest =
UpdateFunctionConfigurationRequest.builder()
                .functionName(functionName)
                .handler(handler)
                .runtime(Runtime.JAVA17)
                .build();

            awsLambda.updateFunctionConfiguration(configurationRequest);

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateFunctionConfiguration을 참조하세요.

시나리오

사진을 관리하기 위한 서버리스 애플리케이션 만들기

다음 코드 예시에서는 사용자가 레이블을 사용하여 사진을 관리할 수 있는 서버리스 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 AWS 커뮤니티의 게시물을 참조하세요.

이 예제에서 사용되는 서비스

- API Gateway
- DynamoDB

- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

고객 피드백 분석을 위한 애플리케이션 생성

다음 코드 예제에서는 고객 의견 카드를 분석하고, 원어에서 번역하고, 감정을 파악하고, 번역된 텍스트에서 오디오 파일을 생성하는 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

이 예제 애플리케이션은 고객 피드백 카드를 분석하고 저장합니다. 특히 뉴욕시에 있는 가상 호텔의 필요를 충족합니다. 호텔은 다양한 언어의 고객들로부터 물리적인 의견 카드의 형태로 피드백을 받습니다. 피드백은 웹 클라이언트를 통해 앱에 업로드됩니다. 의견 카드의 이미지가 업로드된 후 다음 단계가 수행됩니다.

- Amazon Textract를 사용하여 이미지에서 텍스트가 추출됩니다.
- Amazon Comprehend가 추출된 텍스트와 해당 언어의 감정을 파악합니다.
- 추출된 텍스트는 Amazon Translate를 사용하여 영어로 번역됩니다.
- Amazon Polly가 추출된 텍스트에서 오디오 파일을 합성합니다.

전체 앱은 AWS CDK를 사용하여 배포할 수 있습니다. 소스 코드와 배포 지침은 GitHub의 프로젝트를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

API Gateway를 사용하여 Lambda 함수 간접 호출

다음 코드 예제에서는 Amazon API Gateway에서 호출한 AWS Lambda 함수를 생성하는 방법을 보여줍니다.

SDK for Java 2.x

Lambda Java 런타임 API를 사용하여 AWS Lambda 함수를 생성하는 방법을 보여줍니다. 이 예제에서는 다양한 AWS 서비스를 호출하여 특정 사용 사례를 수행합니다. 이 예제에서는 Amazon API Gateway에서 간접 호출한 Lambda 함수를 생성하여 작업 기념일에 대한 Amazon DynamoDB 테이블을 스캔하고 Amazon Simple Notification Service(Amazon SNS)를 사용하여 직원에게 1주년 기념일을 축하하는 문자 메시지를 전송하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

Step Functions를 사용하여 Lambda 함수 간접 호출

다음 코드 예제에서는 AWS Lambda 함수를 순차적으로 호출하는 AWS Step Functions 상태 시스템을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

AWS Step Functions 및를 사용하여 AWS 서버리스 워크플로를 생성하는 방법을 보여줍니다 AWS SDK for Java 2.x. 각 워크플로 단계는 AWS Lambda 함수를 사용하여 구현됩니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- DynamoDB
- Lambda
- Amazon SES
- 단계 함수

예약된 이벤트를 사용하여 Lambda 함수 간접 호출

다음 코드 예제에서는 Amazon EventBridge 예약 이벤트에서 호출된 AWS Lambda 함수를 생성하는 방법을 보여줍니다.

SDK for Java 2.x

AWS Lambda 함수를 호출하는 Amazon EventBridge 예약 이벤트를 생성하는 방법을 보여줍니다. Lambda 함수가 간접 호출될 때 cron 표현식을 사용하여 일정을 예약하도록 EventBridge를 구성합니다. 이 예제에서는 Lambda Java 런타임 API를 사용하여 Lambda 함수를 생성합니다. 이 예제에서는 다양한 AWS 서비스를 호출하여 특정 사용 사례를 수행합니다. 이 예제에서는 1주년 기념일에 직원에게 축하하는 모바일 문자 메시지를 전송하는 앱을 생성하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Neptune API를 사용하여 그래프 데이터 쿼리

다음 코드 예제에서는 Neptune API를 사용하여 그래프 데이터를 쿼리하는 방법을 보여줍니다.

SDK for Java 2.x

Amazon Neptune Java API를 사용하여 VPC 내에서 그래프 데이터를 쿼리하는 Lambda 함수를 생성하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Lambda
- Neptune

서버리스 예제

Lambda 함수를 사용하여 Amazon RDS 데이터베이스에 연결

다음 코드 예제는 RDS 데이터베이스에 연결하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 간단한 데이터베이스 요청을 하고 결과를 반환합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Java를 사용하여 Lambda 함수에서 Amazon RDS 데이터베이스에 연결

```
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();
    }
}
```

Kinesis 트리거에서 간접적으로 Lambda 함수 간접 호출

다음 코드 예제에서는 Kinesis 스트림에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 Kinesis 페이로드를 검색하고, Base64에서 디코딩하고, 레코드 콘텐츠를 로깅합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Java를 사용하여 Lambda에서 Kinesis 이벤트를 사용합니다.

```
// 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;
    }

}
```

DynamoDB 트리거에서 간접적으로 Lambda 함수 간접 호출

다음 코드 예제에서는 DynamoDB 스트림에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 DynamoDB 페이로드를 검색하고 레코드 콘텐츠를 로깅합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Java를 사용하여 Lambda로 DynamoDB 이벤트 소비

```
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()));
    }
}
```

Amazon DocumentDB 트리거에서 간접적으로 Lambda 함수 호출

다음 코드 예제에서는 DocumentDB 변경 스트림에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 DocumentDB 페이로드를 검색하고 레코드 콘텐츠를 로깅합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Java를 사용하여 Lambda로 Amazon DocumentDB 이벤트 소비

```
import java.util.List;
import java.util.Map;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
```

```
public class Example implements RequestHandler<Map<String, Object>, String> {

    @SuppressWarnings("unchecked")
    @Override
    public String handleRequest(Map<String, Object> event, Context context) {
        List<Map<String, Object>> events = (List<Map<String, Object>>)
 event.get("events");
        for (Map<String, Object> record : events) {
            Map<String, Object> eventData = (Map<String, Object>)
 record.get("event");
            processEventData(eventData);
        }

        return "OK";
    }

    @SuppressWarnings("unchecked")
    private void processEventData(Map<String, Object> eventData) {
        String operationType = (String) eventData.get("operationType");
        System.out.println("operationType: %s".formatted(operationType));

        Map<String, Object> ns = (Map<String, Object>) eventData.get("ns");

        String db = (String) ns.get("db");
        System.out.println("db: %s".formatted(db));
        String coll = (String) ns.get("coll");
        System.out.println("coll: %s".formatted(coll));

        Map<String, Object> fullDocument = (Map<String, Object>)
 eventData.get("fullDocument");
        System.out.println("fullDocument: %s".formatted(fullDocument));
    }

}
```

Amazon MSK 트리거를 사용하여 Lambda 함수 간접 호출

다음 코드 예제에서는 Amazon MSK 클러스터에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 MSK 페이로드를 검색하고 레코드 콘텐츠를 로깅합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Java를 사용하여 Lambda로 Amazon MSK 이벤트를 사용합니다.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KafkaEvent;
import com.amazonaws.services.lambda.runtime.events.KafkaEvent.KafkaEventRecord;

import java.util.Base64;
import java.util.Map;

public class Example implements RequestHandler<KafkaEvent, Void> {

    @Override
    public Void handleRequest(KafkaEvent event, Context context) {
        for (Map.Entry<String, java.util.List<KafkaEventRecord>> entry :
 event.getRecords().entrySet()) {
            String key = entry.getKey();
            System.out.println("Key: " + key);

            for (KafkaEventRecord record : entry.getValue()) {
                System.out.println("Record: " + record);

                byte[] value = Base64.getDecoder().decode(record.getValue());
                String message = new String(value);
                System.out.println("Message: " + message);
            }
        }

        return null;
    }
}
```

Amazon S3 트리거를 사용하여 Lambda 함수 간접 호출

다음 코드 예제는 S3 버킷에 객체를 업로드하여 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 해당 함수는 이벤트 파라미터에서 S3 버킷 이름과 객체 키를 검색하고 Amazon S3 API를 호출하여 객체의 콘텐츠 유형을 검색하고 로깅합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Java를 사용하여 Lambda로 S3 이벤트를 사용합니다.

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

Amazon SNS 트리거를 사용하여 Lambda 함수 간접 호출

다음 코드 예제에서는 SNS 주제의 메시지를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 함수는 이벤트 파라미터에서 메시지를 검색하고 각 메시지의 내용을 로깅합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Java를 사용하여 Lambda로 SNS 이벤트를 사용합니다.

```
// 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 트리거에서 간접적으로 Lambda 함수 간접 호출

다음 코드 예제는 SQS 대기열에서 메시지를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 함수는 이벤트 파라미터에서 메시지를 검색하고 각 메시지의 내용을 로깅합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Java를 사용하여 Lambda로 SQS 이벤트 사용

```
// 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;
```

```
        }

    }
}
```

Kinesis 트리거로 Lambda 함수에 대한 배치 항목 실패 보고

다음 코드 예제는 Kinesis 스트림에서 이벤트를 수신하는 Lambda 함수에 대한 부분 배치 응답을 구현하는 방법을 보여줍니다. 이 함수는 응답으로 배치 항목 실패를 보고하고 나중에 해당 메시지를 다시 시도하도록 Lambda에 신호를 보냅니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Java를 사용하여 Lambda로 Kinesis 배치 항목 실패 보고.

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

DynamoDB 트리거로 Lambda 함수에 대한 배치 항목 실패 보고

다음 코드 예제에서는 DynamoDB 스트림에서 이벤트를 수신하는 Lambda 함수에 대한 부분 배치 응답을 구현하는 방법을 보여줍니다. 이 함수는 응답으로 배치 항목 실패를 보고하고 나중에 해당 메시지를 다시 시도하도록 Lambda에 신호를 보냅니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Java를 사용하여 Lambda로 DynamoDB 배치 항목 실패 보고.

```
// 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.util.ArrayList;
import java.util.List;

public class ProcessDynamodbRecords implements RequestHandler<DynamodbEvent,
 StreamsEventResponse> {

    @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 SQS 트리거로 Lambda 함수에 대한 배치 항목 실패 보고

다음 코드 예제는 SQS 대기열에서 이벤트를 수신하는 Lambda 함수에 대한 부분 배치 응답을 구현하는 방법을 보여줍니다. 이 함수는 응답으로 배치 항목 실패를 보고하고 나중에 해당 메시지를 다시 시도하도록 Lambda에 신호를 보냅니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Java를 사용하여 Lambda로 SQS 배치 항목 실패 보고

```
// 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>();

        for (SQSEvent.SQSMessage message : sqsEvent.getRecords()) {
            try {
                //process your message
            } catch (Exception e) {
                //Add failed message identifier to the batchItemFailures list
                batchItemFailures.add(new
 SQSBatchResponse.BatchItemFailure(message.getMessageId()));
            }
        }
```

```
            return new SQSBatchResponse(batchItemFailures);
        }
    }
```

AWS 커뮤니티 기여

서버리스 애플리케이션 빌드 및 테스트

다음 코드 예제에서는 Lambda 및 DynamoDB와 함께 API 게이트웨이를 사용하여 서버리스 애플리케이션을 빌드하고 테스트하는 방법을 보여줍니다.

SDK for Java 2.x

Java SDK를 사용하여 Lambda 및 DynamoDB가 포함된 API 게이트웨이로 구성된 서버리스 애플리케이션을 빌드하고 테스트하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda

SDK for Java 2.x를 사용한 Amazon Lex 예제

다음 코드 예제에서는 Amazon Lex와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시나리오

시나리오

Amazon Lex 챗봇 구축

다음 코드 예제에서는 챗봇을 만들어 웹사이트 방문자를 참여시키는 방법을 보여줍니다.

SDK for Java 2.x

Amazon Lex API를 사용하여 웹 애플리케이션 내에 챗봇을 구축하여 웹 사이트 방문자의 참여를 유도하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

SDK for Java 2.x를 사용한 Amazon Location 예제

다음 코드 예제에서는 Amazon Location과 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시작하기
- 기본 사항
- 작업

시작하기

Amazon Location 시작

다음 코드 예제에서는 Amazon Location Service 사용을 시작하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including 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 need to create a collection using the AWS Management
 * console. For information, see the following documentation.
 *
 * https://docs.aws.amazon.com/location/latest/developerguide/geofence-gs.html

 */
public class HelloLocation {

    private static LocationAsyncClient locationAsyncClient;
    private static final Logger logger =
 LoggerFactory.getLogger(HelloLocation.class);

    // This Singleton pattern ensures that only one `LocationClient`
    // instance.
    private static LocationAsyncClient getClient() {
        if (locationAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
```

```
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            locationAsyncClient = LocationAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return locationAsyncClient;
    }

    public static void main(String[] args) {
        final String usage = """

            Usage:
                <collectionName>

            Where:
                collectionName - The Amazon location collection name.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionName = args[0];
        listGeofences(collectionName);
    }

    /**
     * Lists geofences from a specified geofence collection asynchronously.
     *
     * @param collectionName The name of the geofence collection to list geofences
from.
     * @return A {@link CompletableFuture} representing the result of the
asynchronous operation.
```

```
     *         The future completes when all geofences have been processed and
 logged.
     */
    public static CompletableFuture<Void> listGeofences(String collectionName) {
        ListGeofencesRequest geofencesRequest = ListGeofencesRequest.builder()
                .collectionName(collectionName)
                .build();

        ListGeofencesPublisher paginator =
 getClient().listGeofencesPaginator(geofencesRequest);
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            if (response.entries().isEmpty()) {
                logger.info("No Geofences were found in the collection.");
            } else {
                response.entries().forEach(geofence ->
                        logger.info("Geofence ID: " + geofence.geofenceId())
                );
            }
        });
        return future;
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - ListGeofenceCollections
  - ListGeofences

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- Amazon Location 맵을 만듭니다.
- Amazon Location API 키를 만듭니다.
- 맵 URL을 표시합니다.
- 지오펜스 컬렉션을 만듭니다.
- 지오펜스 지오메트리를 저장합니다.
- 트래커 리소스를 만듭니다.

- 디바이스의 위치를 업데이트합니다.
- 지정된 디바이스에 대한 최신 위치 업데이트를 검색합니다.
- 경로 계산기를 만듭니다.
- 시애틀과 밴쿠버 사이의 거리를 결정합니다.
- Amazon Location 상위 수준 API를 사용합니다.
- Amazon Location 자산을 삭제합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon Location Service 기능을 시연하는 대화형 시나리오를 실행합니다.

```
/*
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 */
public class LocationScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    private static final Logger logger =
 LoggerFactory.getLogger(LocationScenario.class);
    static Scanner scanner = new Scanner(System.in);

    static LocationActions locationActions = new LocationActions();

    public static void main(String[] args) {
        final String usage = """
```

```
            Usage:    <mapName> <keyName> <collectionName> <geoId> <trackerName>
<calculatorName> <deviceId>

            Where:
              mapName - The name of the map to be create (e.g., "AWSMap").
              keyName - The name of the API key to create (e.g., "AWSApiKey").
              collectionName - The name of the geofence collection (e.g.,
"AWSLocationCollection").
              geoId - The geographic identifier used for the geofence or map (e.g.,
"geoId").
              trackerName - The name of the tracker (e.g., "geoTracker").
              calculatorName - The name of the route calculator (e.g.,
"AWSRouteCalc").
              deviceId - The ID of the device (e.g., "iPhone-112356").
            """;

        if (args.length != 7) {
            logger.info(usage);
            return;
        }

        String mapName = args[0];
        String keyName = args[1];
        String collectionName = args[2];
        String geoId = args[3];
        String trackerName = args[4];
        String calculatorName = args[5];
        String deviceId = args[6];

        logger.info("""
            AWS Location Service is a fully managed service offered by Amazon Web
Services (AWS) that
            provides location-based services for developers. This service simplifies
            the integration of location-based features into applications, making it
            easier to build and deploy location-aware applications.

            The AWS Location Service offers a range of location-based services,
            including:

            Maps: The service provides access to high-quality maps, satellite
imagery,\s
            and geospatial data from various providers, allowing developers to\s
            easily embed maps into their applications.
```

```
            Tracking: The Location Service enables real-time tracking of mobile
devices,\s
            assets, or other entities, allowing developers to build applications\s
            that can monitor the location of people, vehicles, or other objects.

            Geocoding: The service provides the ability to convert addresses or\s
            location names into geographic coordinates (latitude and longitude),\s
            and vice versa, enabling developers to integrate location-based search\s
            and routing functionality into their applications.
            """);
        waitForInputToContinue(scanner);
        try {
            runScenario(mapName, keyName, collectionName, geoId, trackerName,
calculatorName, deviceId);
        } catch (RuntimeException e) {
            // Clean up AWS Resources.
            cleanUp(mapName, keyName, collectionName, trackerName, calculatorName);
            logger.info(e.getMessage());
        }
    }

    public static void runScenario(String mapName, String keyName, String
collectionName, String geoId, String trackerName, String calculatorName, String
deviceId) {
        logger.info(DASHES);
        logger.info("1. Create a map");
        logger.info("""
             An AWS Location map can enhance the user experience of your
             application by providing accurate and personalized location-based
             features. For example, you could use the geocoding capabilities to
             allow users to search for and locate businesses, landmarks, or
             other points of interest within a specific region.
            """);

        waitForInputToContinue(scanner);
        String mapArn;
        try {
            mapArn = locationActions.createMap(mapName).join();
            logger.info("The Map ARN is: {}", mapArn);  // Log success in calling
code
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ServiceQuotaExceededException) {
```

```
                logger.error("The request exceeded the maximum quota: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred while creating the map.",
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Create an AWS Location API key");
        logger.info("""
            When you embed a map in a web app or website, the API key is
            included in the map tile URL to authenticate requests. You can
            restrict API keys to specific AWS Location operations (e.g., only
            maps, not geocoding). API keys can expire, ensuring temporary
            access control.
            """);

        try {
            String keyArn = locationActions.createKey(keyName, mapArn).join();
            logger.info("The API key was successfully created: {}", keyArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof AccessDeniedException) {
                logger.error("Request was denied: {}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred while creating the API
key.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Display Map URL");
        logger.info("""
            In order to get the MAP URL, you need to get the API Key value.
            You can get the key value using the AWS Management Console under
            Location Services. This operation cannot be completed using the
            AWS SDK. For more information about getting the key value, see
```

```
            the AWS Location Documentation.
            """);
        String mapUrl = "https://maps.geo.aws.amazon.com/maps/v0/maps/"+mapName+"/
tiles/{z}/{x}/{y}?key={KeyValue}";
        logger.info("Embed this URL in your Web app: " + mapUrl);
        logger.info("");
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Create a geofence collection, which manages and stores
 geofences.");
        waitForInputToContinue(scanner);
        try {
            String collectionArn =
 locationActions.createGeofenceCollection(collectionName).join();
            logger.info("The geofence collection was successfully created: {}",
 collectionArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ConflictException) {
                logger.error("A conflict occurred: {}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred while creating the
 geofence collection.", cause);
            }
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Store a geofence geometry in a given geofence collection.");
        logger.info("""
            An AWS Location geofence is a virtual boundary that defines a geographic
 area
            on a map. It is a useful feature for tracking the location of
            assets or monitoring the movement of objects within a specific region.

            To define a geofence, you need to specify the coordinates of a
            polygon that represents the area of interest. The polygon must be
            defined in a counter-clockwise direction, meaning that the points of
            the polygon must be listed in a counter-clockwise order.
```

```
            This is a requirement for the AWS Location service to correctly
            interpret the geofence and ensure that the location data is
            accurately processed within the defined area.
            """);

        waitForInputToContinue(scanner);
        try {
            locationActions.putGeofence(collectionName, geoId).join();
            logger.info("Successfully created geofence: {}", geoId);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ValidationException) {
                logger.error("A validation error occurred while creating geofence:
{}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("6. Create a tracker resource which lets you retrieve current
and historical location of devices..");
        waitForInputToContinue(scanner);
        try {
            String trackerArn = locationActions.createTracker(trackerName).join();
            logger.info("Successfully created tracker. ARN: {}", trackerArn);  //
Log success
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ConflictException) {
                logger.error("A conflict occurred while creating the tracker: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);
```

```
        logger.info(DASHES);
        logger.info("7. Update the position of a device in the location tracking
system.");
        logger.info("""
            The AWS location service does not enforce a strict format for deviceId,
but it must:
              - Be a string (case-sensitive).
              - Be 1-100 characters long.
              - Contain only:
                - Alphanumeric characters (A-Z, a-z, 0-9)
                - Underscores (_)
                - Hyphens (-)
                - Be the same ID used when sending and retrieving positions.
            """);

        waitForInputToContinue(scanner);
        try {
            CompletableFuture<BatchUpdateDevicePositionResponse> future =
locationActions.updateDevicePosition(trackerName, deviceId);
            future.join();
            logger.info(deviceId + " was successfully updated in the location
tracking system.");
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The resource was not found: {}", cause.getMessage(),
cause);
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("8. Retrieve the most recent position update for a specified
device..");
        waitForInputToContinue(scanner);
        try {
            GetDevicePositionResponse response =
locationActions.getDevicePosition(trackerName, deviceId).join();
```

```
            logger.info("Successfully fetched device position: {}",
response.position());
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The resource was not found: {}", cause.getMessage(),
cause);
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("9. Create a route calculator.");
        waitForInputToContinue(scanner);
        try {
            CreateRouteCalculatorResponse response =
locationActions.createRouteCalculator(calculatorName).join();
            logger.info("Route calculator created successfully: {}",
response.calculatorArn());
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ConflictException) {
                logger.info("A conflict occurred: {}", cause.getMessage(), cause);
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("10. Determine the distance between Seattle and Vancouver using
the route calculator.");
        waitForInputToContinue(scanner);
        try {
            CalculateRouteResponse response =
locationActions.calcDistanceAsync(calculatorName).join();
```

```
            logger.info("Successfully calculated route. The distance in kilometers
is {}", response.summary().distance());
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The resource was not found: {}", cause.getMessage(),
cause);
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("11. Use the GeoPlacesAsyncClient to perform additional
operations.");
        logger.info("""
            This scenario will show use of the GeoPlacesClient that enables
            location search and geocoding capabilities for your applications.\s

            We are going to use this client to perform these AWS Location tasks:
             - Reverse Geocoding (reverseGeocode): Converts geographic coordinates
into addresses.
             - Place Search (searchText): Finds places based on search queries.
             - Nearby Search (searchNearby): Finds places near a specific location.
            """);

        logger.info("First we will perform a Reverse Geocoding operation");
        waitForInputToContinue(scanner);
        try {
            locationActions.reverseGeocode().join();
            logger.info("Now we are going to perform a text search using coffee
shop.");
            waitForInputToContinue(scanner);
            locationActions.searchText("coffee shop").join();
            waitForInputToContinue(scanner);

            logger.info("Now we are going to perform a nearby Search.");
            //waitForInputToContinue(scanner);
            locationActions.searchNearBy().join();
            waitForInputToContinue(scanner);
        } catch (CompletionException ce) {
```

```
            Throwable cause = ce.getCause();
            if (cause instanceof
software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                logger.error("A validation error occurred: {}", cause.getMessage(),
cause);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        logger.info(DASHES);

        logger.info("12. Delete the AWS Location Services resources.");
        logger.info("Would you like to delete the AWS Location Services resources?
(y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            cleanUp(mapName, keyName, collectionName, trackerName, calculatorName);
        } else {
            logger.info("The AWS resources will not be deleted.");
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info(" This concludes the AWS Location Service scenario.");
        logger.info(DASHES);
    }

    /**
     * Cleans up resources by deleting the specified map, key, geofence collection,
tracker, and route calculator.
     *
     * @param mapName The name of the map to delete.
     * @param keyName The name of the key to delete.
     * @param collectionName The name of the geofence collection to delete.
     * @param trackerName The name of the tracker to delete.
     * @param calculatorName The name of the route calculator to delete.
     */
    private static void cleanUp(String mapName, String keyName, String
collectionName, String trackerName, String calculatorName) {
        try {
            locationActions.deleteMap(mapName).join();
```

```
            locationActions.deleteKey(keyName).join();
            locationActions.deleteGeofenceCollectionAsync(collectionName).join();
            locationActions.deleteTracker(trackerName).join();
            locationActions.deleteRouteCalculator(calculatorName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The resource was not found: {}", cause.getMessage(),
 cause);
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage(),
 cause);
            }
            return;
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Amazon Location Service SDK 메서드의 래퍼 클래스입니다.

```
public class LocationActions {

    private static LocationAsyncClient locationAsyncClient;

    private static GeoPlacesAsyncClient geoPlacesAsyncClient;
```

```
    private static final Logger logger =
LoggerFactory.getLogger(LocationActions.class);

    // This Singleton pattern ensures that only one `LocationClient`
    // instance is used throughout the application.
    private LocationAsyncClient getClient() {
        if (locationAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            locationAsyncClient = LocationAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return locationAsyncClient;
    }

    private static GeoPlacesAsyncClient getGeoPlacesClient() {
        if (geoPlacesAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();
```

```
            geoPlacesAsyncClient = GeoPlacesAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return geoPlacesAsyncClient;
    }


    /**
     * Performs a nearby places search based on the provided geographic coordinates
 (latitude and longitude).
     * The method sends an asynchronous request to search for places within a 1-
kilometer radius of the specified location.
     * The results are processed and printed once the search completes successfully.
     */
    public CompletableFuture<SearchNearbyResponse> searchNearBy() {
        double latitude = 37.7749;  // San Francisco
        double longitude = -122.4194;
        List<Double> queryPosition = List.of(longitude, latitude);

        // Set up the request for searching nearby places.
        SearchNearbyRequest request = SearchNearbyRequest.builder()
            .queryPosition(queryPosition)  // Set the position
            .queryRadius(1000L)  // Radius in meters (1000 meters = 1 km).
            .build();

        return getGeoPlacesClient().searchNearby(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof
 software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                        throw new CompletionException("A validation error occurred:
 " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error performing place search",
 exception);
                }

                // Process the response and print the results.
                response.resultItems().forEach(result -> {
                    logger.info("Place Name: " + result.placeType().name());
```

```
                    logger.info("Address: " + result.address().label());
                    logger.info("Distance: " + result.distance() + " meters");
                    logger.info("-------------------------");
                });
            });
    }


    /**
     * Searches for a place using the provided search query and prints the detailed
information of the first result.
     *
     * @param searchQuery the search query to be used for the place search (ex,
coffee shop)
     */
    public CompletableFuture<Void> searchText(String searchQuery) {
        double latitude = 37.7749;  // San Francisco
        double longitude = -122.4194;
        List<Double> queryPosition = List.of(longitude, latitude);

        SearchTextRequest request = SearchTextRequest.builder()
                .queryText(searchQuery)
                .biasPosition(queryPosition)
                .build();

        return getGeoPlacesClient().searchText(request)
                .thenCompose(response -> {
                    if (response.resultItems().isEmpty()) {
                        logger.info("No places found.");
                        return CompletableFuture.completedFuture(null);
                    }

                    // Get the first place ID
                    String placeId = response.resultItems().get(0).placeId();
                    logger.info("Found Place with id: " + placeId);

                    // Fetch detailed info using getPlace
                    GetPlaceRequest getPlaceRequest = GetPlaceRequest.builder()
                            .placeId(placeId)
                            .build();

                    return getGeoPlacesClient().getPlace(getPlaceRequest)
                            .thenAccept(placeResponse -> {
                                logger.info("Detailed Place Information:");
```

```
                                logger.info("Name: " +
 placeResponse.placeType().name());
                                logger.info("Address: " +
 placeResponse.address().label());

                                if (placeResponse.foodTypes() != null && !
placeResponse.foodTypes().isEmpty()) {
                                    logger.info("Food Types:");
                                    placeResponse.foodTypes().forEach(foodType -> {
                                        logger.info("  - " + foodType);
                                    });
                                } else {
                                    logger.info("No food types available.");
                                }
                                logger.info("------------------------");
                            });
                })
                .exceptionally(exception -> {
                    Throwable cause = exception.getCause();
                    if (cause instanceof
 software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                        throw new CompletionException("A validation error occurred:
 " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error performing place search",
 exception);
                });
    }



    /**
     * Performs reverse geocoding using the AWS Geo Places API.
     * Reverse geocoding is the process of converting geographic coordinates
 (latitude and longitude) to a human-readable address.
     * This method uses the latitude and longitude of San Francisco as the input,
 and prints the resulting address.
     */
    public CompletableFuture<ReverseGeocodeResponse> reverseGeocode() {
        double latitude = 37.7749;  // San Francisco
        double longitude = -122.4194;
        logger.info("Use latitude 37.7749 and longitude -122.4194");

        // AWS expects [longitude, latitude].
```

```
        List<Double> queryPosition = List.of(longitude, latitude);
        ReverseGeocodeRequest request = ReverseGeocodeRequest.builder()
            .queryPosition(queryPosition)
            .build();
        CompletableFuture<ReverseGeocodeResponse> futureResponse =
            getGeoPlacesClient().reverseGeocode(request);

        return futureResponse.whenComplete((response, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof
software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                    throw new CompletionException("A validation error occurred: " +
cause.getMessage(), cause);
                }
                throw new CompletionException("Error performing reverse geocoding",
exception);
            }

            response.resultItems().forEach(result ->
                logger.info("The address is: " + result.address().label())
            );
        });
    }


    /**
     * Calculates the distance between two locations asynchronously.
     *
     * @param routeCalcName the name of the route calculator to use
     * @return a {@link CompletableFuture} that will complete with a {@link
CalculateRouteResponse} containing the distance and estimated duration of the route
     */
    public CompletableFuture<CalculateRouteResponse> calcDistanceAsync(String
routeCalcName) {
        // Define coordinates for Seattle, WA and Vancouver, BC.
        List<Double> departurePosition = Arrays.asList(-122.3321, 47.6062);
        List<Double> arrivePosition = Arrays.asList(-123.1216, 49.2827);

        CalculateRouteRequest request = CalculateRouteRequest.builder()
            .calculatorName(routeCalcName)
            .departurePosition(departurePosition)
            .destinationPosition(arrivePosition)
            .travelMode("Car") // Options: Car, Truck, Walking, Bicycle
```

```
            .distanceUnit("Kilometers") // Options: Meters, Kilometers, Miles
            .build();

        return getClient().calculateRoute(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The AWS resource was not
found: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Failed to calculate route: " +
exception.getMessage(), exception);
                }
            });
    }


    /**
     * Creates a new route calculator with the specified name and data source.
     *
     * @param routeCalcName the name of the route calculator to be created
     */
    public CompletableFuture<CreateRouteCalculatorResponse>
createRouteCalculator(String routeCalcName) {
        String dataSource = "Esri"; // or "Here"
        CreateRouteCalculatorRequest request =
CreateRouteCalculatorRequest.builder()
            .calculatorName(routeCalcName)
            .dataSource(dataSource)
            .build();

        return getClient().createRouteCalculator(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflict error occurred: "
+ cause.getMessage(), cause);
                    }
                    throw new CompletionException("Failed to create route
calculator: " + exception.getMessage(), exception);
                }
            });
```

```
    }


    /**
     * Retrieves the position of a device using the provided LocationClient.
     *
     * @param trackerName The name of the tracker associated with the device.
     * @param deviceId    The ID of the device to retrieve the position for.
     * @throws RuntimeException If there is an error fetching the device position.
     */
    public CompletableFuture<GetDevicePositionResponse> getDevicePosition(String
trackerName, String deviceId) {
        GetDevicePositionRequest request = GetDevicePositionRequest.builder()
            .trackerName(trackerName)
            .deviceId(deviceId)
            .build();

        return getClient().getDevicePosition(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The AWS resource was not
found: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error fetching device position: "
+ exception.getMessage(), exception);
                }
            });
    }


    /**
     * Updates the position of a device in the location tracking system.
     *
     * @param trackerName the name of the tracker associated with the device
     * @param deviceId    the unique identifier of the device
     * @throws RuntimeException if an error occurs while updating the device
position
     */
    public CompletableFuture<BatchUpdateDevicePositionResponse>
updateDevicePosition(String trackerName, String deviceId) {
        double latitude = 37.7749;  // Example: San Francisco
        double longitude = -122.4194;
```

```
        DevicePositionUpdate positionUpdate = DevicePositionUpdate.builder()
            .deviceId(deviceId)
            .sampleTime(Instant.now()) // Timestamp of position update.
            .position(Arrays.asList(longitude, latitude)) // AWS requires
[longitude, latitude]
            .build();

        BatchUpdateDevicePositionRequest request =
BatchUpdateDevicePositionRequest.builder()
            .trackerName(trackerName)
            .updates(positionUpdate)
            .build();

        CompletableFuture<BatchUpdateDevicePositionResponse> futureResponse =
getClient().batchUpdateDevicePosition(request);
        return futureResponse.whenComplete((response, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    throw new CompletionException("The resource was not found: " +
cause.getMessage(), cause);
                } else {
                    throw new CompletionException("Error updating device position: "
+ exception.getMessage(), exception);
                }
            }
        });
    }


    /**
     * Creates a new tracker resource in your AWS account, which you can use to
track the location of devices.
     *
     * @param trackerName the name of the tracker to be created
     * @return a {@link CompletableFuture} that, when completed, will contain the
Amazon Resource Name (ARN) of the created tracker
     */
    public CompletableFuture<String> createTracker(String trackerName) {
        CreateTrackerRequest trackerRequest = CreateTrackerRequest.builder()
            .description("Created using the Java V2 SDK")
            .trackerName(trackerName)
```

```
            .positionFiltering("TimeBased") // Options: TimeBased, DistanceBased,
AccuracyBased
            .build();

        return getClient().createTracker(trackerRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("Conflict occurred while
creating tracker: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error creating tracker: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(CreateTrackerResponse::trackerArn); // Return only the
tracker ARN
    }



    /**
     * Adds a new geofence to the specified collection.
     *
     * @param collectionName the name of the geofence collection to add the geofence
to
     * @param geoId          the unique identifier for the geofence
     */
    public CompletableFuture<PutGeofenceResponse> putGeofence(String collectionName,
String geoId) {
        // Define the geofence geometry (polygon).
        GeofenceGeometry geofenceGeometry = GeofenceGeometry.builder()
            .polygon(List.of(
                List.of(
                    List.of(-122.3381, 47.6101), // First point
                    List.of(-122.3281, 47.6101),
                    List.of(-122.3281, 47.6201),
                    List.of(-122.3381, 47.6201),
                    List.of(-122.3381, 47.6101) // Closing the polygon
                )
            ))
            .build();
```

```
        PutGeofenceRequest geofenceRequest = PutGeofenceRequest.builder()
            .collectionName(collectionName) // Specify the collection.
            .geofenceId(geoId) // Unique ID for the geofence.
            .geometry(geofenceGeometry)
            .build();

        return getClient().putGeofence(geofenceRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ValidationException) {
                        throw new CompletionException("Validation error while
creating geofence: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error creating geofence: " +
exception.getMessage(), exception);
                }
            });
    }


    /**
     * Creates a new geofence collection.
     *
     * @param collectionName the name of the geofence collection to be created
     */
    public CompletableFuture<String> createGeofenceCollection(String collectionName)
{
        CreateGeofenceCollectionRequest collectionRequest =
CreateGeofenceCollectionRequest.builder()
            .collectionName(collectionName)
            .description("Created by using the AWS SDK for Java")
            .build();

        return getClient().createGeofenceCollection(collectionRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("The geofence collection was
not created due to ConflictException.", cause);
                    }
                    throw new CompletionException("Failed to create geofence
collection: " + exception.getMessage(), exception);
```

```
                }
            })
            .thenApply(response -> response.collectionArn()); // Return only the ARN
    }




    /**
     * Creates a new API key with the specified name and restrictions.
     *
     * @param keyName the name of the API key to be created
     * @param mapArn  the Amazon Resource Name (ARN) of the map resource to which
the API key will be associated
     * @return a {@link CompletableFuture} that completes with the Amazon Resource
Name (ARN) of the created API key,
     * or {@code null} if the operation failed
     */
    public CompletableFuture<String> createKey(String keyName, String mapArn) {
        ApiKeyRestrictions keyRestrictions = ApiKeyRestrictions.builder()
            .allowActions("geo:GetMap*")
            .allowResources(mapArn)
            .build();

        CreateKeyRequest request = CreateKeyRequest.builder()
            .keyName(keyName)
            .restrictions(keyRestrictions)
            .noExpiry(true)
            .build();

        return getClient().createKey(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof AccessDeniedException) {
                        throw new CompletionException("The request was denied
because of insufficient access or permissions.", cause);
                    }
                    throw new CompletionException("Failed to create API key: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.keyArn()); // This will never return
null if the response reaches here
```

```
    }



    /**
     * Creates a new map with the specified name and configuration.
     *
     * @param mapName the name of the map to be created
     * @return a {@link CompletableFuture} that, when completed, will contain the
Amazon Resource Name (ARN) of the created map
     * @throws CompletionException if an error occurs while creating the map, such
as exceeding the service quota
     */
    public CompletableFuture<String> createMap(String mapName) {
        MapConfiguration configuration = MapConfiguration.builder()
            .style("VectorEsriNavigation")
            .build();

        CreateMapRequest mapRequest = CreateMapRequest.builder()
            .mapName(mapName)
            .configuration(configuration)
            .description("A map created using the Java V2 API")
            .build();

        return getClient().createMap(mapRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ServiceQuotaExceededException) {
                        throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                    }
                    throw new CompletionException("Failed to create map: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.mapArn()); // Return the map ARN
    }



    /**
     * Deletes a geofence collection asynchronously.
     *
```

```
    * @param collectionName the name of the geofence collection to be deleted
    * @return a {@link CompletableFuture} that completes when the geofence
collection has been deleted
    */
   public CompletableFuture<Void> deleteGeofenceCollectionAsync(String
collectionName) {
       DeleteGeofenceCollectionRequest collectionRequest =
DeleteGeofenceCollectionRequest.builder()
           .collectionName(collectionName)
           .build();

       return getClient().deleteGeofenceCollection(collectionRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The requested geofence
collection was not found.", cause);
                   }
                   throw new CompletionException("Failed to delete geofence
collection: " + exception.getMessage(), exception);
               }
               logger.info("The geofence collection {} was deleted.",
collectionName);
           })
           .thenApply(response -> null);
   }



   /**
    * Deletes the specified key from the key-value store.
    *
    * @param keyName the name of the key to be deleted
    * @return a {@link CompletableFuture} that completes when the key has been
deleted
    * @throws CompletionException if the key was not found or if an error occurred
during the deletion process
    */
   public CompletableFuture<Void> deleteKey(String keyName) {
       DeleteKeyRequest keyRequest = DeleteKeyRequest.builder()
           .keyName(keyName)
           .build();
```

```
        return getClient().deleteKey(keyRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The key was not found.",
cause);
                    }
                    throw new CompletionException("Failed to delete key: " +
exception.getMessage(), exception);
                }
                logger.info("The key {} was deleted.", keyName);
            })
            .thenApply(response -> null);
    }


    /**
     * Deletes a map with the specified name.
     *
     * @param mapName the name of the map to be deleted
     * @return a {@link CompletableFuture} that completes when the map deletion is
successful, or throws a {@link CompletionException} if an error occurs
     */
    public CompletableFuture<Void> deleteMap(String mapName) {
        DeleteMapRequest mapRequest = DeleteMapRequest.builder()
            .mapName(mapName)
            .build();

        return getClient().deleteMap(mapRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The map was not found.",
cause);
                    }
                    throw new CompletionException("Failed to delete map: " +
exception.getMessage(), exception);
                }
                logger.info("The map {} was deleted.", mapName);
            })
            .thenApply(response -> null);
    }
```

```
   /**
    * Deletes a tracker with the specified name.
    *
    * @param trackerName the name of the tracker to be deleted
    * @return a {@link CompletableFuture} that completes when the tracker has been
deleted
    * @throws CompletionException if an error occurs while deleting the tracker
    *                             - if the tracker was not found, a {@link
ResourceNotFoundException} is thrown wrapped in the CompletionException
    *                             - if any other error occurs, a generic
CompletionException is thrown with the error message
    */
   public CompletableFuture<Void> deleteTracker(String trackerName) {
       DeleteTrackerRequest trackerRequest = DeleteTrackerRequest.builder()
           .trackerName(trackerName)
           .build();

       return getClient().deleteTracker(trackerRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The tracker was not found.",
cause);
                   }
                   throw new CompletionException("Failed to delete the tracker: " +
exception.getMessage(), exception);
               }
               logger.info("The tracker {} was deleted.", trackerName);
           })
           .thenApply(response -> null); // Ensures CompletableFuture<Void>
   }


   /**
    * Deletes a route calculator from the system.
    *
    * @param calcName the name of the route calculator to delete
    * @return a {@link CompletableFuture} that completes when the route calculator
has been deleted
    * @throws CompletionException if an error occurs while deleting the route
calculator
```

```
     *                                - If the route calculator was not found, a {@link
 ResourceNotFoundException} will be thrown
     *                                - If any other error occurs, a generic {@link
 CompletionException} will be thrown
     */
    public CompletableFuture<Void> deleteRouteCalculator(String calcName) {
        DeleteRouteCalculatorRequest calculatorRequest =
 DeleteRouteCalculatorRequest.builder()
            .calculatorName(calcName)
            .build();

        return getClient().deleteRouteCalculator(calculatorRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The route calculator was not
 found.", cause);
                    }
                    throw new CompletionException("Failed to delete the route
 calculator: " + exception.getMessage(), exception);
                }
                logger.info("The route calculator {} was deleted.", calcName);
            })
            .thenApply(response -> null);
    }
}
```

작업

`BatchUpdateDevicePosition`

다음 코드 예시는 `BatchUpdateDevicePosition`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Updates the position of a device in the location tracking system.
    *
    * @param trackerName the name of the tracker associated with the device
    * @param deviceId    the unique identifier of the device
    * @throws RuntimeException if an error occurs while updating the device
position
    */
   public CompletableFuture<BatchUpdateDevicePositionResponse>
updateDevicePosition(String trackerName, String deviceId) {
       double latitude = 37.7749;  // Example: San Francisco
       double longitude = -122.4194;

       DevicePositionUpdate positionUpdate = DevicePositionUpdate.builder()
           .deviceId(deviceId)
           .sampleTime(Instant.now()) // Timestamp of position update.
           .position(Arrays.asList(longitude, latitude)) // AWS requires
[longitude, latitude]
           .build();

       BatchUpdateDevicePositionRequest request =
BatchUpdateDevicePositionRequest.builder()
           .trackerName(trackerName)
           .updates(positionUpdate)
           .build();

       CompletableFuture<BatchUpdateDevicePositionResponse> futureResponse =
getClient().batchUpdateDevicePosition(request);
       return futureResponse.whenComplete((response, exception) -> {
           if (exception != null) {
               Throwable cause = exception.getCause();
               if (cause instanceof ResourceNotFoundException) {
                   throw new CompletionException("The resource was not found: " +
cause.getMessage(), cause);
               } else {
                   throw new CompletionException("Error updating device position: "
+ exception.getMessage(), exception);
               }
           }
       });
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 BatchUpdateDevicePosition을 참조하세요.

`CalculateRoute`

다음 코드 예시는 `CalculateRoute`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Calculates the distance between two locations asynchronously.
    *
    * @param routeCalcName the name of the route calculator to use
    * @return a {@link CompletableFuture} that will complete with a {@link
CalculateRouteResponse} containing the distance and estimated duration of the route
    */
   public CompletableFuture<CalculateRouteResponse> calcDistanceAsync(String
routeCalcName) {
       // Define coordinates for Seattle, WA and Vancouver, BC.
       List<Double> departurePosition = Arrays.asList(-122.3321, 47.6062);
       List<Double> arrivePosition = Arrays.asList(-123.1216, 49.2827);

       CalculateRouteRequest request = CalculateRouteRequest.builder()
           .calculatorName(routeCalcName)
           .departurePosition(departurePosition)
           .destinationPosition(arrivePosition)
           .travelMode("Car") // Options: Car, Truck, Walking, Bicycle
           .distanceUnit("Kilometers") // Options: Meters, Kilometers, Miles
           .build();

       return getClient().calculateRoute(request)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
```

```
                        throw new CompletionException("The AWS resource was not
found: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Failed to calculate route: " +
exception.getMessage(), exception);
                }
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CalculateRoute를 참조하세요.

`CreateGeofenceCollection`

다음 코드 예시는 `CreateGeofenceCollection`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Creates a new geofence collection.
     *
     * @param collectionName the name of the geofence collection to be created
     */
    public CompletableFuture<String> createGeofenceCollection(String collectionName)
{
        CreateGeofenceCollectionRequest collectionRequest =
CreateGeofenceCollectionRequest.builder()
            .collectionName(collectionName)
            .description("Created by using the AWS SDK for Java")
            .build();

        return getClient().createGeofenceCollection(collectionRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
```

```
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("The geofence collection was
not created due to ConflictException.", cause);
                    }
                    throw new CompletionException("Failed to create geofence
collection: " + exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.collectionArn()); // Return only the ARN
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateGeofenceCollection을 참조하세요.

`CreateKey`

다음 코드 예시는 `CreateKey`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Creates a new API key with the specified name and restrictions.
     *
     * @param keyName the name of the API key to be created
     * @param mapArn  the Amazon Resource Name (ARN) of the map resource to which
the API key will be associated
     * @return a {@link CompletableFuture} that completes with the Amazon Resource
Name (ARN) of the created API key,
     * or {@code null} if the operation failed
     */
    public CompletableFuture<String> createKey(String keyName, String mapArn) {
        ApiKeyRestrictions keyRestrictions = ApiKeyRestrictions.builder()
            .allowActions("geo:GetMap*")
```

```
            .allowResources(mapArn)
            .build();

        CreateKeyRequest request = CreateKeyRequest.builder()
            .keyName(keyName)
            .restrictions(keyRestrictions)
            .noExpiry(true)
            .build();

        return getClient().createKey(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof AccessDeniedException) {
                        throw new CompletionException("The request was denied
because of insufficient access or permissions.", cause);
                    }
                    throw new CompletionException("Failed to create API key: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.keyArn()); // This will never return
null if the response reaches here
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateKey를 참조하세요.

`CreateMap`

다음 코드 예시는 `CreateMap`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Creates a new map with the specified name and configuration.
     *
     * @param mapName the name of the map to be created
     * @return a {@link CompletableFuture} that, when completed, will contain the
Amazon Resource Name (ARN) of the created map
     * @throws CompletionException if an error occurs while creating the map, such
as exceeding the service quota
     */
    public CompletableFuture<String> createMap(String mapName) {
        MapConfiguration configuration = MapConfiguration.builder()
            .style("VectorEsriNavigation")
            .build();

        CreateMapRequest mapRequest = CreateMapRequest.builder()
            .mapName(mapName)
            .configuration(configuration)
            .description("A map created using the Java V2 API")
            .build();

        return getClient().createMap(mapRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ServiceQuotaExceededException) {
                        throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                    }
                    throw new CompletionException("Failed to create map: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.mapArn()); // Return the map ARN
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateMap을 참조하세요.

`CreateRouteCalculator`

다음 코드 예시는 `CreateRouteCalculator`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Creates a new route calculator with the specified name and data source.
    *
    * @param routeCalcName the name of the route calculator to be created
    */
   public CompletableFuture<CreateRouteCalculatorResponse>
createRouteCalculator(String routeCalcName) {
       String dataSource = "Esri"; // or "Here"
       CreateRouteCalculatorRequest request =
CreateRouteCalculatorRequest.builder()
           .calculatorName(routeCalcName)
           .dataSource(dataSource)
           .build();

       return getClient().createRouteCalculator(request)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ConflictException) {
                       throw new CompletionException("A conflict error occurred: "
+ cause.getMessage(), cause);
                   }
                   throw new CompletionException("Failed to create route
calculator: " + exception.getMessage(), exception);
               }
           });
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateRouteCalculator를 참조하세요.

`CreateTracker`

다음 코드 예시는 `CreateTracker`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Creates a new tracker resource in your AWS account, which you can use to
track the location of devices.
    *
    * @param trackerName the name of the tracker to be created
    * @return a {@link CompletableFuture} that, when completed, will contain the
Amazon Resource Name (ARN) of the created tracker
    */
   public CompletableFuture<String> createTracker(String trackerName) {
       CreateTrackerRequest trackerRequest = CreateTrackerRequest.builder()
           .description("Created using the Java V2 SDK")
           .trackerName(trackerName)
           .positionFiltering("TimeBased") // Options: TimeBased, DistanceBased,
AccuracyBased
           .build();

       return getClient().createTracker(trackerRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ConflictException) {
                       throw new CompletionException("Conflict occurred while
creating tracker: " + cause.getMessage(), cause);
                   }
                   throw new CompletionException("Error creating tracker: " +
exception.getMessage(), exception);
               }
           })
           .thenApply(CreateTrackerResponse::trackerArn); // Return only the
tracker ARN
```

```
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateTracker를 참조하세요.

`DeleteGeofenceCollection`

다음 코드 예시는 `DeleteGeofenceCollection`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Deletes a geofence collection asynchronously.
    *
    * @param collectionName the name of the geofence collection to be deleted
    * @return a {@link CompletableFuture} that completes when the geofence
collection has been deleted
    */
   public CompletableFuture<Void> deleteGeofenceCollectionAsync(String
collectionName) {
       DeleteGeofenceCollectionRequest collectionRequest =
DeleteGeofenceCollectionRequest.builder()
           .collectionName(collectionName)
           .build();

       return getClient().deleteGeofenceCollection(collectionRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The requested geofence
collection was not found.", cause);
                   }
```

```
                    throw new CompletionException("Failed to delete geofence
collection: " + exception.getMessage(), exception);
                }
                logger.info("The geofence collection {} was deleted.",
collectionName);
            })
            .thenApply(response -> null);
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteGeofenceCollection을 참조하세요.

`DeleteKey`

다음 코드 예시는 `DeleteKey`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Deletes the specified key from the key-value store.
     *
     * @param keyName the name of the key to be deleted
     * @return a {@link CompletableFuture} that completes when the key has been
deleted
     * @throws CompletionException if the key was not found or if an error occurred
during the deletion process
     */
    public CompletableFuture<Void> deleteKey(String keyName) {
        DeleteKeyRequest keyRequest = DeleteKeyRequest.builder()
            .keyName(keyName)
            .build();

        return getClient().deleteKey(keyRequest)
            .whenComplete((response, exception) -> {
```

```
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The key was not found.",
cause);
                    }
                    throw new CompletionException("Failed to delete key: " +
exception.getMessage(), exception);
                }
                logger.info("The key {} was deleted.", keyName);
            })
            .thenApply(response -> null);
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteKey를 참조하세요.

`DeleteMap`

다음 코드 예시는 `DeleteMap`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Deletes a map with the specified name.
     *
     * @param mapName the name of the map to be deleted
     * @return a {@link CompletableFuture} that completes when the map deletion is
successful, or throws a {@link CompletionException} if an error occurs
     */
    public CompletableFuture<Void> deleteMap(String mapName) {
        DeleteMapRequest mapRequest = DeleteMapRequest.builder()
            .mapName(mapName)
            .build();
```

```
        return getClient().deleteMap(mapRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The map was not found.",
cause);
                    }
                    throw new CompletionException("Failed to delete map: " +
exception.getMessage(), exception);
                }
                logger.info("The map {} was deleted.", mapName);
            })
            .thenApply(response -> null);
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteMap을 참조하세요.

`DeleteRouteCalculator`

다음 코드 예시는 `DeleteRouteCalculator`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Deletes a route calculator from the system.
     *
     * @param calcName the name of the route calculator to delete
     * @return a {@link CompletableFuture} that completes when the route calculator
has been deleted
     * @throws CompletionException if an error occurs while deleting the route
calculator
     *                             - If the route calculator was not found, a {@link
ResourceNotFoundException} will be thrown
```

```
     *                              - If any other error occurs, a generic {@link
CompletionException} will be thrown
     */
    public CompletableFuture<Void> deleteRouteCalculator(String calcName) {
        DeleteRouteCalculatorRequest calculatorRequest =
DeleteRouteCalculatorRequest.builder()
            .calculatorName(calcName)
            .build();

        return getClient().deleteRouteCalculator(calculatorRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The route calculator was not
found.", cause);
                    }
                    throw new CompletionException("Failed to delete the route
calculator: " + exception.getMessage(), exception);
                }
                logger.info("The route calculator {} was deleted.", calcName);
            })
            .thenApply(response -> null);
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteRouteCalculator를 참조하세요.

`DeleteTracker`

다음 코드 예시는 `DeleteTracker`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Deletes a tracker with the specified name.
     *
     * @param trackerName the name of the tracker to be deleted
     * @return a {@link CompletableFuture} that completes when the tracker has been
deleted
     * @throws CompletionException if an error occurs while deleting the tracker
     *                             - if the tracker was not found, a {@link
ResourceNotFoundException} is thrown wrapped in the CompletionException
     *                             - if any other error occurs, a generic
CompletionException is thrown with the error message
     */
    public CompletableFuture<Void> deleteTracker(String trackerName) {
        DeleteTrackerRequest trackerRequest = DeleteTrackerRequest.builder()
            .trackerName(trackerName)
            .build();

        return getClient().deleteTracker(trackerRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The tracker was not found.",
cause);
                    }
                    throw new CompletionException("Failed to delete the tracker: " +
exception.getMessage(), exception);
                }
                logger.info("The tracker {} was deleted.", trackerName);
            })
            .thenApply(response -> null); // Ensures CompletableFuture<Void>
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteTracker를 참조하세요.

GetDevicePosition

다음 코드 예시는 GetDevicePosition의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Retrieves the position of a device using the provided LocationClient.
    *
    * @param trackerName The name of the tracker associated with the device.
    * @param deviceId    The ID of the device to retrieve the position for.
    * @throws RuntimeException If there is an error fetching the device position.
    */
   public CompletableFuture<GetDevicePositionResponse> getDevicePosition(String
trackerName, String deviceId) {
       GetDevicePositionRequest request = GetDevicePositionRequest.builder()
           .trackerName(trackerName)
           .deviceId(deviceId)
           .build();

       return getClient().getDevicePosition(request)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The AWS resource was not
found: " + cause.getMessage(), cause);
                   }
                   throw new CompletionException("Error fetching device position: "
+ exception.getMessage(), exception);
               }
           });
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetDevicePosition을 참조하세요.

PutGeofence

다음 코드 예시는 PutGeofence의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Adds a new geofence to the specified collection.
    *
    * @param collectionName the name of the geofence collection to add the geofence
to
    * @param geoId          the unique identifier for the geofence
    */
   public CompletableFuture<PutGeofenceResponse> putGeofence(String collectionName,
String geoId) {
       // Define the geofence geometry (polygon).
       GeofenceGeometry geofenceGeometry = GeofenceGeometry.builder()
           .polygon(List.of(
               List.of(
                   List.of(-122.3381, 47.6101), // First point
                   List.of(-122.3281, 47.6101),
                   List.of(-122.3281, 47.6201),
                   List.of(-122.3381, 47.6201),
                   List.of(-122.3381, 47.6101) // Closing the polygon
               )
           ))
           .build();

       PutGeofenceRequest geofenceRequest = PutGeofenceRequest.builder()
           .collectionName(collectionName) // Specify the collection.
           .geofenceId(geoId) // Unique ID for the geofence.
           .geometry(geofenceGeometry)
           .build();

       return getClient().putGeofence(geofenceRequest)
           .whenComplete((response, exception) -> {
```

```
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ValidationException) {
                        throw new CompletionException("Validation error while
creating geofence: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error creating geofence: " +
exception.getMessage(), exception);
                }
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutGeofence를 참조하세요.

SDK for Java 2.x를 사용한 Location Service Places 예제

다음 코드 예제에서는 Location Service Places와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`ReverseGeocode`

다음 코드 예시는 `ReverseGeocode`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Performs reverse geocoding using the AWS Geo Places API.
    * Reverse geocoding is the process of converting geographic coordinates
(latitude and longitude) to a human-readable address.
    * This method uses the latitude and longitude of San Francisco as the input,
and prints the resulting address.
    */
   public CompletableFuture<ReverseGeocodeResponse> reverseGeocode() {
       double latitude = 37.7749;  // San Francisco
       double longitude = -122.4194;
       logger.info("Use latitude 37.7749 and longitude -122.4194");

       // AWS expects [longitude, latitude].
       List<Double> queryPosition = List.of(longitude, latitude);
       ReverseGeocodeRequest request = ReverseGeocodeRequest.builder()
           .queryPosition(queryPosition)
           .build();
       CompletableFuture<ReverseGeocodeResponse> futureResponse =
           getGeoPlacesClient().reverseGeocode(request);

       return futureResponse.whenComplete((response, exception) -> {
           if (exception != null) {
               Throwable cause = exception.getCause();
               if (cause instanceof
software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                   throw new CompletionException("A validation error occurred: " +
cause.getMessage(), cause);
               }
               throw new CompletionException("Error performing reverse geocoding",
exception);
           }

           response.resultItems().forEach(result ->
```

```
                logger.info("The address is: " + result.address().label())
            );
        });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ReverseGeocode를 참조하세요.

`SearchNearby`

다음 코드 예시는 `SearchNearby`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Performs a nearby places search based on the provided geographic coordinates
 (latitude and longitude).
     * The method sends an asynchronous request to search for places within a 1-
kilometer radius of the specified location.
     * The results are processed and printed once the search completes successfully.
     */
    public CompletableFuture<SearchNearbyResponse> searchNearBy() {
        double latitude = 37.7749;  // San Francisco
        double longitude = -122.4194;
        List<Double> queryPosition = List.of(longitude, latitude);

        // Set up the request for searching nearby places.
        SearchNearbyRequest request = SearchNearbyRequest.builder()
            .queryPosition(queryPosition)  // Set the position
            .queryRadius(1000L)  // Radius in meters (1000 meters = 1 km).
            .build();

        return getGeoPlacesClient().searchNearby(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
```

```
                    Throwable cause = exception.getCause();
                    if (cause instanceof
software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                        throw new CompletionException("A validation error occurred:
" + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error performing place search",
exception);
                }

                // Process the response and print the results.
                response.resultItems().forEach(result -> {
                    logger.info("Place Name: " + result.placeType().name());
                    logger.info("Address: " + result.address().label());
                    logger.info("Distance: " + result.distance() + " meters");
                    logger.info("-------------------------");
                });
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 SearchNearby를 참조하세요.

`SearchText`

다음 코드 예시는 `SearchText`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Searches for a place using the provided search query and prints the detailed
information of the first result.
     *
     * @param searchQuery the search query to be used for the place search (ex,
coffee shop)
```

```
     */
    public CompletableFuture<Void> searchText(String searchQuery) {
        double latitude = 37.7749;  // San Francisco
        double longitude = -122.4194;
        List<Double> queryPosition = List.of(longitude, latitude);

        SearchTextRequest request = SearchTextRequest.builder()
                .queryText(searchQuery)
                .biasPosition(queryPosition)
                .build();

        return getGeoPlacesClient().searchText(request)
                .thenCompose(response -> {
                    if (response.resultItems().isEmpty()) {
                        logger.info("No places found.");
                        return CompletableFuture.completedFuture(null);
                    }

                    // Get the first place ID
                    String placeId = response.resultItems().get(0).placeId();
                    logger.info("Found Place with id: " + placeId);

                    // Fetch detailed info using getPlace
                    GetPlaceRequest getPlaceRequest = GetPlaceRequest.builder()
                            .placeId(placeId)
                            .build();

                    return getGeoPlacesClient().getPlace(getPlaceRequest)
                            .thenAccept(placeResponse -> {
                                logger.info("Detailed Place Information:");
                                logger.info("Name: " +
 placeResponse.placeType().name());
                                logger.info("Address: " +
 placeResponse.address().label());

                                if (placeResponse.foodTypes() != null && !
placeResponse.foodTypes().isEmpty()) {
                                    logger.info("Food Types:");
                                    placeResponse.foodTypes().forEach(foodType -> {
                                        logger.info("  - " + foodType);
                                    });
                                } else {
                                    logger.info("No food types available.");
                                }
```

```
                                logger.info("-------------------------");
                            });
                })
                .exceptionally(exception -> {
                    Throwable cause = exception.getCause();
                    if (cause instanceof
software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                        throw new CompletionException("A validation error occurred:
" + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error performing place search",
exception);
                });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 SearchText를 참조하세요.

AWS Marketplace SDK for Java 2.x를 사용한 카탈로그 API 예제

다음 코드 예제에서는 AWS Marketplace Catalog API와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- AMI 제품
- 채널 파트너 제안
- 컨테이너 제품
- 개체
- 제안
- Products
- 재판매 권한 부여
- SaaS 제품
- 유틸리티

AMI 제품

기존 AMI 제품에 차원 추가 및 제안 요금 조건 업데이트

다음 코드 예제에서는 기존 AMI 제품에 차원을 추가하고 제안 요금 조건을 업데이트하는 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 `RunChangesets`에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Identifier": "prod-1111111111111",
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": [
                {
                    "Key": "m7g.8xlarge",
                    "Description": "m7g.8xlarge",
                    "Name": "m7g.8xlarge",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "m7g.4xlarge",
                                        "Price": "0.45"
                                    },
                                    {
                                        "DimensionKey": "m7g.2xlarge",
                                        "Price": "0.45"
                                    },
                                    {
                                        "DimensionKey": "m7g.8xlarge",
                                        "Price": "0.55"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

AMI 제품이 배포되는 리전 추가

다음 코드 예제에서는 AMI 제품이 배포되는 리전을 추가하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 `RunChangesets`에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-2",
                    "us-west-2"
                ]
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

공개 또는 제한된 AMI 제품 및 시간 단위 연간 요금이 청구되는 공개 제안 생성

다음 코드 예제에서는 공개 또는 제한된 AMI 제품 및 시간 단위 연간 요금이 청구되는 공개 제안을 생성하는 방법을 보여줍니다. 이 예제에서는 표준 또는 사용자 지정 EULA를 생성합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 `RunChangesets`에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
```

```
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "InstanceTypes": [
                    "t2.micro"
                ]
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
                "DeliveryOptions": [
                    {
                        "Details": {
```

```
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-1111111111111111",
                                    "AccessRoleArn":
"arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
                                    }
                                ]
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "t2.micro",
                    "Description": "t2.micro",
                    "Name": "t2.micro",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
```

```
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace API
Reference Code",
```

```
                "Description": "Test public offer with hourly-annual pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P365D"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
```

```
                                "QuantityConfiguration": "Allowed"
                            }
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateLegalTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "LegalTerm",
                    "Documents": [
                        {
                            "Type": "StandardEula",
                            "Version": "2022-07-14"
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateSupportTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "SupportTerm",
                    "RefundPolicy": "Absolutely no refund, period."
                }
            ]
        }
    },
    {
```

```
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

공개 또는 제한된 AMI 제품 및 시간 단위 월간 요금이 청구되는 공개 제안 생성

다음 코드 예제에서는 공개 또는 제한된 AMI 제품 및 시간 단위 월간 요금이 청구되는 공개 제안을 생성하는 방법을 보여줍니다. 이 예제에서는 표준 또는 사용자 지정 EULA를 생성합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 `RunChangesets`에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
```

```
            "Type": "AmiProduct@1.0",
            "Identifier": "$CreateProductChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "ProductTitle": "Sample product",
            "ShortDescription": "Brief description",
            "LongDescription": "Detailed description",
            "Highlights": [
                "Sample highlight"
            ],
            "SearchKeywords": [
                "Sample keyword"
            ],
            "Categories": [
                "Operating Systems"
            ],
            "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
            "VideoUrls": [
                "https://sample.amazonaws.com/awsmp-video-1"
            ],
            "AdditionalResources": []
        }
    },
    {
        "ChangeType": "AddRegions",
        "Entity": {
            "Type": "AmiProduct@1.0",
            "Identifier": "$CreateProductChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Regions": [
                "us-east-1"
            ]
        }
    },
    {
        "ChangeType": "AddInstanceTypes",
        "Entity": {
            "Type": "AmiProduct@1.0",
            "Identifier": "$CreateProductChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "InstanceTypes": [
                "t2.micro"
```

```
                ]
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
                "DeliveryOptions": [
                    {
                        "Details": {
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-1111111111111111",
                                    "AccessRoleArn":
"arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
                                    }
                                ]
                            }
                        }
                    }
                ]
            }
```

```
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "t2.micro",
                    "Description": "t2.micro",
                    "Name": "t2.micro",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
```

```
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with hourly-monthly pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    },
```

```
                {
                    "Type": "RecurringPaymentTerm",
                    "CurrencyCode": "USD",
                    "BillingPeriod": "Monthly",
                    "Price": "15.0"
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateLegalTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "LegalTerm",
                    "Documents": [
                        {
                            "Type": "StandardEula",
                            "Version": "2022-07-14"
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateSupportTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "SupportTerm",
                    "RefundPolicy": "Absolutely no refund, period."
                }
            ]
        }
    },
```

```
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

공개 또는 제한된 AMI 제품 및 시간 단위 요금이 청구되는 공개 제안 생성

다음 코드 예제에서는 공개 또는 제한된 AMI 제품 및 시간 단위 요금이 청구되는 공개 제안을 생성하는 방법을 보여줍니다. 이 예제에서는 표준 또는 사용자 지정 EULA를 생성합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
```

```
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "InstanceTypes": [
```

```
                    "t2.micro"
                ]
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
                "DeliveryOptions": [
                    {
                        "Details": {
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-11111111111111111",
                                    "AccessRoleArn":
"arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
                                    }
                                ]
                            }
                        }
                    }
                ]
```

```
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "t2.micro",
                    "Description": "t2.micro",
                    "Name": "t2.micro",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
```

```
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with hourly pricing for AmiProduct
using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
```

```
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateLegalTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "LegalTerm",
                    "Documents": [
                        {
                            "Type": "StandardEula",
                            "Version": "2022-07-14"
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateSupportTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "SupportTerm",
                    "RefundPolicy": "Absolutely no refund, period."
                }
            ]
        }
    },
    {
        "ChangeType": "ReleaseOffer",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

공개 제안 초안을 사용하여 AMI 제품 초안 생성

다음 코드 예제에서는 공개 제안 초안으로 AMI 제품 초안을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 `RunChangesets`에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product"
            }
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
```

```
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier",
                "Name": "Test Offer"
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

AMI 제품이 배포되는 리전 제한

다음 코드 예제에서는 AMI 제품이 배포되는 리전을 제한하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 `RunChangesets`에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "RestrictRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-west-2"
                ]
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

제품 가시성 제한

다음 코드 예제에서는 제품 가시성을 제한하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 `RunChangesets`에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateVisibility",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "TargetVisibility": "Restricted"
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

AMI 자산을 새 리전에 배포할지 여부 지정

다음 코드 예제에서는 향후 리전을 지원하기 AWS 위해에서 빌드한 새 리전에 AMI 자산을 배포할지 여부를 지정하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 `RunChangesets`에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateFutureRegionSupport",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "FutureRegionSupport": {
                    "SupportedRegions": [
                        "All"
                    ]
                }
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

채널 파트너 제안

모든 제품 유형에 대한 CPPO 초안 생성

다음 코드 예제에서는 구매자를 대상으로 게시하기 전에 내부적으로 검토할 수 있도록 모든 제품 유형에 대한 CPPO 초안을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 `RunChangesets`에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOfferUsingResaleAuthorization",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ResaleAuthorizationId": "11111111-1111-1111-1111-111111111111",
                "Name": "Test Offer",
                "Description": "Test product"
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

계약 요금으로 재판매 권한 부여 대체 비공개 제안 생성

다음 코드 예제에서는 계약 요금이 적용되는 기존 계약에서 재판매 권한 부여 대체 비공개 제안을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 `RunChangesets`에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateReplacementOfferUsingResaleAuthorization",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateReplacementOfferResaleAuth",
            "DetailsDocument": {
                "AgreementId": "agmt-11111111111111111111111",
                "ResaleAuthorizationId": "resaleauthz-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test replacement offer for SaaSProduct using AWS
 Marketplace API Reference Codes",
                "Description": "Test private resale replacement offer with contract
 pricing for SaaSProduct"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Duration": "P12M",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 2
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementEndDate": "2024-01-30"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
```

```
                    "Type": "PaymentScheduleTerm",
                    "CurrencyCode": "USD",
                    "Schedule": [
                        {
                            "ChargeDate": "2024-01-01",
                            "ChargeAmount": "0"
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateLegalTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "LegalTerm",
                    "Documents": [
                        {
                            "Type": "StandardEula",
                            "Version": "2022-07-14"
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateAvailability",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
        },
        "DetailsDocument": {
            "AvailabilityEndDate": "2023-12-31"
        }
    },
    {
        "ChangeType": "ReleaseOffer",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

채널 파트너가 생성한 모든 CPPO 나열

다음 코드 예제에서는 채널 파트너가 생성한 모든 CPPO를 나열하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
```

```
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;

public class ListAllCppoOffers {

 /*
  * List all CPPOs created by a channel partner
  */
 public static void main(String[] args) {

  List<String> cppoOfferIds = getAllCppoOfferIds();

  ReferenceCodesUtils.formatOutput(cppoOfferIds);
 }

 public static List<String> getAllCppoOfferIds() {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  // get all offer entity ids
  List<String> entityIdList = new ArrayList<String>();

  ListEntitiesRequest listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_OFFER)
    .maxResults(10)
    .nextToken(null)
    .build();

  ListEntitiesResponse listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);

  for (EntitySummary entitySummary : listEntitiesResponse.entitySummaryList()) {
   entityIdList.add(entitySummary.entityId());
  }

  while (listEntitiesResponse.nextToken() != null) {
   listEntitiesRequest =
     ListEntitiesRequest.builder()
     .catalog(AWS_MP_CATALOG)
```

```
      .entityType(ENTITY_TYPE_OFFER)
      .maxResults(10)
      .nextToken(listEntitiesResponse.nextToken())
      .build();
    listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);

    for (EntitySummary entitySummary : listEntitiesResponse.entitySummaryList()) {
     entityIdList.add(entitySummary.entityId());
    }
  }

  // filter for CPPO offers: ResaleAuthorizationId exists in Details

  List<String> cppoOfferIds = new ArrayList<String>();

  for (String entityId : entityIdList) {
   DescribeEntityRequest describeEntityRequest =
     DescribeEntityRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityId(entityId)
     .build();
    DescribeEntityResponse describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);

    Document resaleAuthorizationDocument =
 describeEntityResponse.detailsDocument().asMap().get(ATTRIBUTE_RESALE_AUTHORIZATION_ID);
    String resaleAuthorizationId = resaleAuthorizationDocument != null ?
 resaleAuthorizationDocument.asString() : "";

    if (!resaleAuthorizationId.isEmpty()) {
        cppoOfferIds.add(resaleAuthorizationId);
    }
  }
  return cppoOfferIds;
 }

}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListEntities를 참조하세요.

채널 파트너가 사용할 수 있는 모든 공유 재판매 권한 나열

다음 코드 예제에서는 채널 파트너가 사용할 수 있는 모든 공유 재판매 권한을 나열하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;

public class ListAllSharedResaleAuthorizations {

 /*
  * list all resale authorizations shared to an account
  */
 public static void main(String[] args) {

  List<ListEntitiesResponse> responseList = getListEntityResponseList();
  ReferenceCodesUtils.formatOutput(responseList);
 }

 public static List<ListEntitiesResponse> getListEntityResponseList() {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
```

```
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  List<ListEntitiesResponse> responseList = new ArrayList<ListEntitiesResponse>();

  ListEntitiesRequest listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_RESALE_AUTHORIZATION)
    .maxResults(10)
    .ownershipType(OWNERSHIP_TYPE_SHARED)
    .nextToken(null)
    .build();

  ListEntitiesResponse listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);

  responseList.add(listEntitiesResponse);

  while (listEntitiesResponse.nextToken() != null) {
   listEntitiesRequest = ListEntitiesRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityType(ENTITY_TYPE_RESALE_AUTHORIZATION)
     .maxResults(10)
     .ownershipType(OWNERSHIP_TYPE_SHARED)
     .nextToken(listEntitiesResponse.nextToken())
     .build();

   listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);

   responseList.add(listEntitiesResponse);
  }
  return responseList;
 }

}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListEntities를 참조하세요.

CPPO 게시 및 구매자 EULA 추가

다음 코드 예제에서는 CPPO를 게시하고 구매자 EULA를 추가하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 `RunChangesets`에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPOoffer",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111",
                "Name": "Test Offer",
                "Description":"Test product"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
           {
               "Type": "LegalTerm",
               "Documents": [
                {
                 "Type": "CustomEula",
```

```
                "Url": "https://s3.amazonaws.com/sample-bucket/custom-eula.pdf"
              }
             ]
          }
       ]
        }
       },
       {
           "ChangeType": "UpdateTargeting",
           "Entity": {
               "Type": "Offer@1.0",
               "Identifier": "$CreateCPPOoffer.Entity.Identifier"
           },
           "DetailsDocument": {
               "PositiveTargeting": {
                   "BuyerAccounts": ["222222222222"]
                   }
           }
       },
       {
           "ChangeType": "UpdateAvailability",
           "Entity": {
               "Type": "Offer@1.0",
               "Identifier": "$CreateCPPOoffer.Entity.Identifier"
           },
           "DetailsDocument": {
               "AvailabilityEndDate": "2023-07-31"
               }
       },
       {
           "ChangeType": "UpdateValidityTerms",
           "Entity": {
               "Type": "Offer@1.0",
               "Identifier": "$CreateCPPOoffer.Entity.Identifier"
           },
           "DetailsDocument": {
               "Terms": [
                   {
                       "Type": "ValidityTerm",
                       "AgreementDuration": "P450D"
                   }
               ]
        }
       },
```

```
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

일회성 재판매 권한 부여를 사용한 CPPO 게시 및 가격 인상 업데이트

다음 코드 예제에서는 AMI, SaaS 또는 컨테이너 제품에 대한 일회성 재판매 권한 부여를 사용하여 CPPO를 게시하고 가격 인상을 업데이트하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPOoffer",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111",
                "Name": "Test Offer",
                "Description":"Test product"
```

```
            }
        },
        {
            "ChangeType": "UpdateMarkup",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Percentage" : "5.0"
        }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": ["222222222222"]
                    }
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-07-31"
                }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
```

```
                        "AgreementDuration": "P450D"
                    }
                ]
             }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

CPPO 초안 게시 및 가격 인상 업데이트

다음 코드 예제에서는 초안 CPPO를 게시하고 가격 인상을 업데이트하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 `RunChangesets`에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPOoffer",
```

```
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111",
                "Name": "Test Offer",
                "Description":"Test product"
            }
        },
        {
            "ChangeType": "UpdateMarkup",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Percentage" : "5.0"
        }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": ["222222222222"]
                    }
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-07-31"
                }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P450D"
                    }
                ]
             }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

CPPO의 만료 날짜 업데이트

다음 코드 예제에서는 구매자에게 제안을 평가하고 수락하는 데 필요한 시간을 더 줄 수 있도록 CPPO의 만료 날짜를 업데이트하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
```

```
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2025-07-31"
                }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

컨테이너 제품

공개 제안 초안을 사용하여 컨테이너 제품 초안 생성

다음 코드 예제에서는 공개 제안 초안을 사용하여 컨테이너 제품 초안을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
    "Catalog":"AWSMarketplace",
    "changeSet":[
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "ContainerProduct@1.0"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product"
```

```
            }
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier",
                "Name": "Test Offer"
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

공개 제안 및 계약 요금으로 제한된 컨테이너 제품 생성

다음 코드 예제에서는 공개 제안, 계약 요금, 표준 EULA를 갖춘 제한된 컨테이너 제품을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "ContainerProduct@1.0"
            },
```

```
            "DetailsDocument": {},
            "ChangeName": "CreateProductChange"
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "Categories": [
                    "Streaming solutions"
                ],
                "ProductTitle": "ContainerProduct",
                "AdditionalResources": [],
                "LongDescription": "Long description goes here",
                "SearchKeywords": [
                    "container streaming"
                ],
                "ShortDescription": "Description1",
                "Highlights": [
                    "Highlight 1",
                    "Highlight 2"
                ],
                "SupportDescription": "No support available",
                "VideoUrls": []
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "Cores",
                    "Description": "Cores per cluster",
                    "Name": "Cores",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
```

```
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddRepositories",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Repositories": [
                    {
                        "RepositoryName": "uniquerepositoryname",
                        "RepositoryType": "ECR"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "ContainerProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
```

```
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            },
            "ChangeName": "CreateOfferChange"
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "Constraints": {
                                    "MultipleDimensionSelection": "Disallowed",
                                    "QuantityConfiguration": "Disallowed"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "Cores",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "No refunds"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Some container offer Name",
                "Description": "Some interesting container offer description"
            }
        },
        {
            "ChangeType": "UpdateRenewalTerms",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "RenewalTerm"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

개체

단일 직접 호출로 모든 엔터티 설명

다음 코드 예제에서는 단일 호출로 모든 엔터티를 설명하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.catalogapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.BatchDescribeEntitiesRequest;
import software.amazon.awssdk.services.marketplacecatalog.model.EntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.BatchDescribeEntitiesResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.EntityDetail;
import
 software.amazon.awssdk.services.marketplacecatalog.model.BatchDescribeErrorDetail;

import java.util.Arrays;
import java.util.Map;

public class BatchDescribeEntities {

    /*
     * BatchDescribe my entities in a single call and
     *  check if it contains all the information I need to know about the entities.
     */
    public static void main(String[] args) {

        MarketplaceCatalogClient marketplaceCatalogClient =
                MarketplaceCatalogClient.builder()
                        .httpClient(ApacheHttpClient.builder().build())
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();

        BatchDescribeEntitiesRequest batchDescribeEntitiesRequest =
                BatchDescribeEntitiesRequest.builder()
                        .entityRequestList(Arrays.asList(
                                EntityRequest.builder()
                                        .catalog(AWS_MP_CATALOG).entityId(OFFER_ID)
                                        .build(),
                                EntityRequest.builder()

 .catalog(AWS_MP_CATALOG).entityId(PRODUCT_ID)
```

```
                                        .build()))
                        .build();

        BatchDescribeEntitiesResponse batchDescribeEntitiesResponse =
 marketplaceCatalogClient.batchDescribeEntities(batchDescribeEntitiesRequest);

        // Reading the successful entities response
        Map<String, EntityDetail> entityDetailsMap =
 batchDescribeEntitiesResponse.entityDetails();
        for (Map.Entry<String, EntityDetail> entry : entityDetailsMap.entrySet()) {
            System.out.println("EntityId: " + entry.getKey());
            ReferenceCodesUtils.formatOutput(entry.getValue());
        }

        // Logging the failed entities error details
        Map<String, BatchDescribeErrorDetail> entityErrorsMap =
 batchDescribeEntitiesResponse.errors();
        for (Map.Entry<String, BatchDescribeErrorDetail> entry :
 entityErrorsMap.entrySet()) {
            System.out.println(String.format("EntityId: %s, ErrorCode: %s,
 ErrorMessage: %s", entry.getKey(),
                    entry.getValue().errorCode(), entry.getValue().errorMessage()));
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 BatchDescribeEntities를 참조하세요.

제품과 관련된 모든 제안 나열 및 설명

다음 코드 예제에서는 제품과 관련된 모든 제안을 나열하고 설명하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
package com.example.awsmarketplace.catalogapi;
```

```
import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import software.amazon.awssdk.services.marketplacecatalog.model.EntityTypeFilters;
import software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferProductIdFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferTargetingFilter;

public class ListProductPrivateOffers {

 private static MarketplaceCatalogClient marketplaceCatalogClient =
   MarketplaceCatalogClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();
 /*
  * retrieve all private offer information related to a single product
  */
 public static void main(String[] args) {

  List<EntitySummary> entitySummaryList = getEntitySummaryList();

  // for each offer id, output the offer detail using DescribeEntity API

  for (EntitySummary entitySummary : entitySummaryList) {
   DescribeEntityRequest describeEntityRequest =
     DescribeEntityRequest.builder()
```

```
    .catalog(AWS_MP_CATALOG)
    .entityId(entitySummary.entityId())
    .build();
  DescribeEntityResponse describeEntityResponse =
marketplaceCatalogClient.describeEntity(describeEntityRequest);
  ReferenceCodesUtils.formatOutput(describeEntityResponse);
 }
}
public static List<EntitySummary> getEntitySummaryList() {
 // define list entities filters

 EntityTypeFilters entityTypeFilters =
   EntityTypeFilters.builder()
   .offerFilters(OfferFilters.builder()
     .targeting(OfferTargetingFilter.builder()
       .valueListWithStrings(OFFER_TARGETING_BUYERACCOUNTS)
       .build())
     .productId(OfferProductIdFilter.builder()
       .valueList(PRODUCT_ID)
       .build())
     .build())
   .build();

 ListEntitiesRequest listEntitiesRequest =
   ListEntitiesRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .entityType(ENTITY_TYPE_OFFER).maxResults(50)
   .entityTypeFilters(entityTypeFilters)
   .nextToken(null)
   .build();

 ListEntitiesResponse listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);

 // save all entitySummary of the results into entitySummaryList

 List<EntitySummary> entitySummaryList = new ArrayList<EntitySummary>();

 entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());

 while ( listEntitiesResponse.nextToken() != null &&
listEntitiesResponse.nextToken().length() > 0) {
  listEntitiesRequest =
    ListEntitiesRequest.builder()
```

```
        .catalog(AWS_MP_CATALOG)
        .entityType(ENTITY_TYPE_OFFER).maxResults(50)
        .entityTypeFilters(entityTypeFilters)
        .nextToken(listEntitiesResponse.nextToken())
        .build();
      listEntitiesResponse =
    marketplaceCatalogClient.listEntities(listEntitiesRequest);
      entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());
     }
     return entitySummaryList;
    }

}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - DescribeEntity
  - ListEntities

제안

SaaS 제품에 대한 사용자 지정 차원 생성 및 비공개 제안 생성

다음 코드 예제에서는 SaaS 제품에 대한 사용자 지정 차원을 생성하고 비공개 제안을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
```

```
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": [
                {
                    "Types": [
                        "Entitled"
                    ],
                    "Description": "Custom Pricing 4 w/ terms and coverage to be
defined in Private Offer",
                    "Unit": "Units",
                    "Key": "Custom4",
                    "Name": "Custom Pricing 4"
                }
            ]
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            },
            "ChangeName": "CreateOfferChange"
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Private Test Offer - SaaS Contract Product",
                "Description": "Private Test Offer - SaaS Contract Product"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "Custom4",
                                        "Price": "300.0"
                                    }
                                ],
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P36M"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ],
    "ChangeSetName": "PrivateOfferWithCustomDimension"
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

AMI 또는 SaaS 제품에 대한 비공개 제안 초안 생성

다음 코드 예제에서는 구매자를 대상으로 게시하기 전에 내부적으로 검토할 수 있도록 AMI 또는 SaaS 제품에 대한 비공개 제안 초안을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 `RunChangesets`에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "Test Private Offer"
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

SaaS 제품에 대한 계약 및 종량제 요금이 적용되는 비공개 제안 생성

다음 코드 예제에서는 SaaS 제품에 대한 계약 및 종량제 요금이 적용되는 비공개 제안을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 `RunChangesets`에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
 API Reference Code",
                "Description": "Test private offer with subscription pricing for
 SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
```

```
                                "Value": "P12M"
                            },
                            "RateCard": [
                                {
                                    "DimensionKey": "BasicService",
                                    "Price": "150"
                                },
                                {
                                    "DimensionKey": "PremiumService",
                                    "Price": "300"
                                }
                            ],
                            "Constraints": {
                                "MultipleDimensionSelection": "Allowed",
                                "QuantityConfiguration": "Allowed"
                            }
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateLegalTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "LegalTerm",
                    "Documents": [
                        {
                            "Type": "CustomEula",
                            "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                        }
                    ]
                }
            ]
        }
    },
    {
```

```
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

SaaS 제품에 대한 계약 요금 및 유연한 결제 일정이 적용되는 비공개 제안 생성

다음 코드 예제에서는 SaaS 제품에 대한 계약 요금 및 유연한 결제 일정이 적용되는 비공개 제안을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
```

```
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with subscription pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 1
                            },
                            {
                                "DimensionKey": "PremiumService",
                                "MaxQuantity": 1
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P12M"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
```

```
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "200.00"
                            },
                            {
                                "ChargeDate": "2024-02-01",
                                "ChargeAmount": "170.00"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

컨테이너 제품에 대한 계약 요금이 적용되는 비공개 제안 생성

다음 코드 예제에서는 컨테이너 제품에 대한 계약 요금이 적용되는 비공개 제안을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
```

```
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for Container product using AWS
Marketplace API Reference Code",
                "Description": "Test private offer for Container product with
contract pricing using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
```

```
                            "Selector": {
                                "Type": "Duration",
                                "Value": "P12M"
                            },
                            "Constraints": {
                                "MultipleDimensionSelection": "Disallowed",
                                "QuantityConfiguration": "Disallowed"
                            },
                            "RateCard": [
                                {
                                    "DimensionKey": "ReqPerHour",
                                    "Price": "0.25"
                                }
                            ]
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateLegalTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "LegalTerm",
                    "Documents": [
                        {
                            "Type": "StandardEula",
                            "Version": "2022-07-14"
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateAvailability",
        "Entity": {
            "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

AMI 제품에 대한 계약 요금이 적용되는 비공개 제안 생성

다음 코드 예제에서는 AMI 제품에 대한 계약 요금이 적용되는 비공개 제안을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 `RunChangesets`에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
```

```
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace API
Reference Code",
                "Description": "Test private offer with hourly annual pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
```

```
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "ReadOnlyUsers",
                                        "Price": "220.00"
                                    }
```

```
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

AMI 제품에 대한 시간 단위 연간 요금 및 유연한 결제 일정이 적용되는 비공개 제안 생성

다음 코드 예제에서는 AMI 제품에 대한 시간당 연간 요금과 유연한 결제 일정이 적용되는 비공개 제안을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
```

```
   "ChangeSet": [
       {
           "ChangeType": "CreateOffer",
           "ChangeName": "CreateOfferChange",
           "Entity": {
               "Type": "Offer@1.0"
           },
           "DetailsDocument": {
               "ProductId": "prod-1111111111111"
           }
       },
       {
           "ChangeType": "UpdateInformation",
           "Entity": {
               "Type": "Offer@1.0",
               "Identifier": "$CreateOfferChange.Entity.Identifier"
           },
           "DetailsDocument": {
               "Name": "Test private offer for AmiProduct using AWS Marketplace API
Reference Code",
               "Description": "Test private offer with hourly annual pricing for
AmiProduct using AWS Marketplace API Reference Code"
           }
       },
       {
           "ChangeType": "UpdateTargeting",
           "Entity": {
               "Type": "Offer@1.0",
               "Identifier": "$CreateOfferChange.Entity.Identifier"
           },
           "DetailsDocument": {
               "PositiveTargeting": {
                   "BuyerAccounts": [
                       "111111111111",
                       "222222222222"
                   ]
               }
           }
       },
       {
           "ChangeType": "UpdateLegalTerms",
           "Entity": {
               "Type": "Offer@1.0",
               "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.17"
```

```
                                }
                            ]
                        }
                    ]
                },
                {
                    "Type": "FixedUpfrontPricingTerm",
                    "CurrencyCode": "USD",
                    "Price": "0.0",
                    "Duration": "P365D",
                    "Grants": [
                        {
                            "DimensionKey": "t2.micro",
                            "MaxQuantity": 1
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateValidityTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "ValidityTerm",
                    "AgreementDuration": "P650D"
                }
            ]
        }
    },
    {
        "ChangeType": "UpdatePaymentScheduleTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
```

```
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "200.00"
                            },
                            {
                                "ChargeDate": "2024-02-01",
                                "ChargeAmount": "170.00"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

AMI 제품에 대한 시간 단위 연간 요금이 적용되는 비공개 제안 생성

다음 코드 예제에서는 AMI 제품에 대한 시간당 연간 요금이 적용되는 비공개 제안을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace API
 Reference Code",
                "Description": "Test private offer with hourly annual pricing for
 AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
```

```
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
```

```
                    "RateCards": [
                        {
                            "RateCard": [
                                {
                                    "DimensionKey": "t2.micro",
                                    "Price": "0.17"
                                }
                            ]
                        }
                    ]
                },
                {
                    "Type": "ConfigurableUpfrontPricingTerm",
                    "CurrencyCode": "USD",
                    "RateCards": [
                        {
                            "Selector": {
                                "Type": "Duration",
                                "Value": "P365D"
                            },
                            "RateCard": [
                                {
                                    "DimensionKey": "t2.micro",
                                    "Price": "220.00"
                                }
                            ],
                            "Constraints": {
                                "MultipleDimensionSelection": "Allowed",
                                "QuantityConfiguration": "Allowed"
                            }
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateValidityTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
```

```
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P650D"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

AMI 제품에 대한 시간 단위 요금이 적용되는 비공개 제안 생성

다음 코드 예제에서는 AMI 제품에 대한 시간당 요금이 적용되는 비공개 제안을 생성하는 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
```

```
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace API
Reference Code",
                "Description": "Test private offer with hourly pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
```

```
                    "Type": "LegalTerm",
                    "Documents": [
                        {
                            "Type": "StandardEula",
                            "Version": "2022-07-14"
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateAvailability",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "AvailabilityEndDate": "2025-01-01"
        }
    },
    {
        "ChangeType": "UpdatePricingTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "PricingModel": "Usage",
            "Terms": [
                {
                    "Type": "UsageBasedPricingTerm",
                    "CurrencyCode": "USD",
                    "RateCards": [
                        {
                            "RateCard": [
                                {
                                    "DimensionKey": "t2.micro",
                                    "Price": "0.15"
                                }
                            ]
                        }
                    ]
                }
```

```
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P30D"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

SaaS 제품에 대한 구독 요금이 적용되는 비공개 제안 생성

다음 코드 예제에서는 SaaS 제품에 대한 구독 요금이 적용되는 비공개 제안을 생성하는 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 `RunChangesets`에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
 API Reference Code",
                "Description": "Test private offer with subscription pricing for
 SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.13"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.22"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P30D"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

SaaS 제품에 대한 계층형 계약 요금이 적용되는 비공개 제안 생성

다음 코드 예제에서는 SaaS 제품에 대한 계층형 계약 요금이 적용되는 비공개 제안을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with subscription pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
```

```
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "120.00"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "200.00"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Disallowed",
                                    "QuantityConfiguration": "Disallowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

SaaS 제품에 대한 구독 요금이 적용되는 공개 무료 평가판 제안 생성

다음 코드 예제에서는 SaaS 제품에 대한 구독 요금이 적용되는 공개 무료 평가판 제안을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
```

```
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public free trial offer for SaaSProduct using AWS
Marketplace API Reference Code",
                "Description": "Test public free trial offer with subscription
pricing for SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Free",
                "Terms": [
                    {
                        "Type": "FreeTrialPricingTerm",
                        "Duration": "P20D",
                        "Grants": [
                            {
                                "DimensionKey": "WorkloadSmall"
                            },
                            {
                                "DimensionKey": "WorkloadMedium"
                            }
                        ]
                    }
                ]
            }
        },
```

```
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

계약 요금이 적용되는 대체 비공개 제안 생성

다음 코드 예제에서는 계약 요금이 적용되는 기존 계약에서 대체 비공개 제안을 생성하는 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 `RunChangesets`에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateReplacementOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateReplacementOffer",
            "DetailsDocument": {
                "AgreementId": "agmt-11111111111111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test replacement offer for SaaSProduct using AWS
 Marketplace API Reference Codes",
                "Description": "Test private replacement offer with contract pricing
 for SaaSProduct"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 2
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementEndDate": "2024-01-30"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
```

```
                    "Schedule": [
                        {
                            "ChargeDate": "2024-01-01",
                            "ChargeAmount": "0"
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateLegalTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateReplacementOffer.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "LegalTerm",
                    "Documents": [
                        {
                            "Type": "StandardEula",
                            "Version": "2022-07-14"
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateAvailability",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateReplacementOffer.Entity.Identifier"
        },
        "DetailsDocument": {
            "AvailabilityEndDate": "2023-12-31"
        }
    },
    {
        "ChangeType": "ReleaseOffer",
        "Entity": {
            "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

공개 제안 설명

다음 코드 예제에서는 공개 제안을 설명하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.catalogapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;

public class DescribeEntity {

 /*
  * Describe my AMI or SaaS or Container product and check if it contains all the
 information I need to know about the product
```

```
 */
public static void main(String[] args) {

 String offerId = args.length > 0 ? args[0] : OFFER_ID;

 DescribeEntityResponse describeEntityResponse =
getDescribeEntityResponse(offerId);

 ReferenceCodesUtils.formatOutput(describeEntityResponse);
}

public static DescribeEntityResponse getDescribeEntityResponse(String offerId) {
 MarketplaceCatalogClient marketplaceCatalogClient =
   MarketplaceCatalogClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 DescribeEntityRequest describeEntityRequest =
   DescribeEntityRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .entityId(offerId)
   .build();

 DescribeEntityResponse describeEntityResponse =
marketplaceCatalogClient.describeEntity(describeEntityRequest);
 return describeEntityResponse;
}
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeEntity를 참조하세요.

비공개 제안 초안 만료

다음 코드 예제에서는 구매자가 더 이상 제안을 볼 수 없도록 비공개 제안의 만료 날짜를 과거 날짜로 설정하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 `RunChangesets`에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-01-01"
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

모든 비공개 제안 나열

다음 코드 예제에서는 모든 비공개 제안을 나열하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import software.amazon.awssdk.services.marketplacecatalog.model.EntityTypeFilters;
import software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferAvailabilityEndDateFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferAvailabilityEndDateFilterDate
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferBuyerAccountsFilter;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferReleaseDateFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferReleaseDateFilterDateRange;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferTargetingFilter;

public class ListAllPrivateOffers {

 /*
  * List all my private offers and sort or filter them by Offer Publish Date, Offer
 Expiry Date and Buyer IDs
  *
  * OfferTargetingFilter = BuyerAccounts (private offer);
  * OfferBuyerAccountsFilter: Buyer IDs filter
  * OfferAvailabilityEndDateFilter : Offer Expiry Date filter
```

```
 * OfferReleaseDateFilter : Offer Publish Date filter
 */

private static MarketplaceCatalogClient marketplaceCatalogClient =
  MarketplaceCatalogClient.builder()
  .httpClient(ApacheHttpClient.builder().build())
  .credentialsProvider(ProfileCredentialsProvider.create())
  .build();

public static void main(String[] args) {

 String offerReleaseDateAfterValue = "2023-01-01T23:59:59Z";
 String offerAvailableEndDateAfterValue = "2040-12-24T23:59:59Z";

 List<EntitySummary> entitySummaryList =
getEntitySummaryList(offerReleaseDateAfterValue, offerAvailableEndDateAfterValue);

 // for each offer id, output the offer detail using DescribeEntity API


 for (EntitySummary entitySummary : entitySummaryList) {
  DescribeEntityRequest describeEntityRequest =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(entitySummary.entityId())
    .build();
  DescribeEntityResponse describeEntityResponse =
marketplaceCatalogClient.describeEntity(describeEntityRequest);
  ReferenceCodesUtils.formatOutput(describeEntityResponse);
 }
}

public static List<EntitySummary> getEntitySummaryList (String
offerReleaseDateAfterValue, String offerAvailableEndDateAfterValue) {

 EntityTypeFilters entityTypeFilters =
   EntityTypeFilters.builder()
   .offerFilters(OfferFilters.builder()
     .targeting(OfferTargetingFilter.builder()
       .valueListWithStrings(OFFER_TARGETING_BUYERACCOUNTS)
       .build())
     .buyerAccounts(OfferBuyerAccountsFilter.builder()
       .wildCardValue(BUYER_ACCOUNT_ID)
       .build())
```

```
      .availabilityEndDate(OfferAvailabilityEndDateFilter.builder()
        .dateRange(OfferAvailabilityEndDateFilterDateRange.builder()
          .afterValue(offerAvailableEndDateAfterValue).build())
        .build())
      .releaseDate(OfferReleaseDateFilter.builder()
        .dateRange(OfferReleaseDateFilterDateRange.builder()
          .afterValue(offerReleaseDateAfterValue)
          .build())
        .build())
      .build())
    .build();

 ListEntitiesRequest listEntitiesRequest =
   ListEntitiesRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .entityType(ENTITY_TYPE_OFFER).maxResults(10)
   .entityTypeFilters(entityTypeFilters)
   .nextToken(null)
   .build();

 ListEntitiesResponse listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);
 List<EntitySummary> entitySummaryList = new ArrayList<EntitySummary>();

 entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());

 while ( listEntitiesResponse.nextToken() != null &&
listEntitiesResponse.nextToken().length() > 0) {
  listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_OFFER)
    .maxResults(10)
    .entityTypeFilters(entityTypeFilters)
    .nextToken(listEntitiesResponse.nextToken())
    .build();
  listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);
  entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());
 }

 return entitySummaryList;
}
```

```
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

특정 제품 ID에 대해 릴리스된 공개 및 비공개 제안 나열

다음 코드 예제에서는 특정 제품 ID에 대해 릴리스된 공개 및 비공개 제안을 나열하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import software.amazon.awssdk.services.marketplacecatalog.model.EntityTypeFilters;
import software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferProductIdFilter;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferStateFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferTargetingFilter;

public class ListProductPublicOrPrivateReleasedOffers {
```

```
/*
 * List released Public/Private offers for a specific product id.
 * Example below is to list released public offers.
 * To change to released private offers, change OFFER_TARGETING_NONE (None) to
OFFER_TARGETING_BUYERACCOUNTS(BuyerAccounts)
 */
public static void main(String[] args) {

 List<EntitySummary> entitySummaryList = getEntitySummaryLIst();
 ReferenceCodesUtils.formatOutput(entitySummaryList);
}

public static List<EntitySummary> getEntitySummaryLIst() {
 MarketplaceCatalogClient marketplaceCatalogClient =
   MarketplaceCatalogClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 // define list entities filters

 EntityTypeFilters entityTypeFilters =
   EntityTypeFilters.builder()
   .offerFilters(OfferFilters.builder()
     .targeting(OfferTargetingFilter.builder()
       .valueListWithStrings(OFFER_TARGETING_NONE)
       .build())
     .state(OfferStateFilter.builder()
       .valueListWithStrings(OFFER_STATE_RELEASED)
       .build())
     .productId(OfferProductIdFilter.builder()
       .valueList(PRODUCT_ID)
       .build())
     .build())
   .build();

 ListEntitiesRequest listEntitiesRequest =
   ListEntitiesRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .entityType(ENTITY_TYPE_OFFER)
   .maxResults(10)
   .entityTypeFilters(entityTypeFilters)
   .nextToken(null)
```

```
    .build();

 ListEntitiesResponse listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);

 // save all entitySummary of the results into entitySummaryList

 List<EntitySummary> entitySummaryList = new ArrayList<EntitySummary>();

 entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());

 while ( listEntitiesResponse.nextToken() != null &&
listEntitiesResponse.nextToken().length() > 0) {
  listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_OFFER)
    .maxResults(10)
    .entityTypeFilters(entityTypeFilters)
    .nextToken(listEntitiesResponse.nextToken())
    .build();
  listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);
   entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());
 }
 return entitySummaryList;
}

}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

종량제 요금으로 계약을 적용하도록 제안 업데이트

다음 코드 예제에서는 종량제 요금으로 계약을 적용하도록 제안을 업데이트하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 `RunChangesets`에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    },
                    {
```

```
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

시간 단위 연간 요금을 적용하도록 제안 업데이트

다음 코드 예제에서는 시간당 연간 요금을 적용하도록 제안을 업데이트하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 `RunChangesets`에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "0.13"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
```

```
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P365D"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "20.03"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

특정 지리적 리전을 대상으로 적용하도록 제안 업데이트

다음 코드 예제에서는 특정 지리적 리전을 대상으로 적용하도록 제안을 업데이트하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
```

```
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "CountryCodes": [
                        "US",
                        "ES",
                        "FR",
                        "AU"
                    ]
                }
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

공개 제안의 이름 및 설명 업데이트

다음 코드 예제에서는 공개 제안의 이름과 설명을 업데이트하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
```

```
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

제안의 EULA 업데이트

다음 코드 예제에서는 제안의 EULA를 업데이트하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
```

```
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Name": "New offer name",
                "Description": "New offer description"
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

비공개 제안의 만료 날짜를 미래 날짜로 업데이트

다음 코드 예제에서는 구매자에게 제안을 평가하고 수락할 시간을 더 많이 제공할 수 있도록 비공개 제안의 만료 날짜를 미래 날짜로 업데이트하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 `RunChangesets`에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
```

```
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2026-01-01"
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

SaaS 제품에 대한 공개 무료 평가판 제안의 무료 평가판 기간 업데이트

다음 코드 예제에서는 SaaS 제품에 대한 공개 무료 평가판 제안의 무료 평가판 기간을 업데이트하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 `RunChangesets`에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "FreeTrialPricingTerm",
                        "Duration": "P21D",
                        "Grants": [
```

```
                            {
                                "DimensionKey": "WorkloadSmall"
                            },
                            {
                                "DimensionKey": "WorkloadMedium"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

제품의 환불 정책 업데이트

다음 코드 예제에서는 제안의 환불 정책을 업데이트하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Terms": [
```

```
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Updated refund policy description"
                    }
                ]
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

Products

AMI, SaaS 또는 컨테이너 제품 설명

다음 코드 예제에서는 AMI, SaaS 또는 컨테이너 제품을 설명하고 제품에 대해 알고 싶은 모든 정보가 포함되어 있는지 확인하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.catalogapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;
```

```
public class DescribeEntity {

 /*
  * Describe my AMI or SaaS or Container product and check if it contains all the
 information I need to know about the product
  */
 public static void main(String[] args) {

  String offerId = args.length > 0 ? args[0] : OFFER_ID;

  DescribeEntityResponse describeEntityResponse =
 getDescribeEntityResponse(offerId);

  ReferenceCodesUtils.formatOutput(describeEntityResponse);
 }

 public static DescribeEntityResponse getDescribeEntityResponse(String offerId) {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeEntityRequest describeEntityRequest =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(offerId)
    .build();

  DescribeEntityResponse describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);
  return describeEntityResponse;
 }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeEntity를 참조하세요.

모든 AMI, SaaS 또는 컨테이너 제품 및 관련 공개 제안 나열

다음 코드 예제에서는 모든 AMI, SaaS 또는 컨테이너 제품 및 관련 공개 제안을 나열하는 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import software.amazon.awssdk.services.marketplacecatalog.model.EntityTypeFilters;
import software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferProductIdFilter;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferStateFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferTargetingFilter;

public class ListEntities {

 /*
  * List all my AMI or SaaS or Container products and associated public offers
  */
 public static void main(String[] args) {

  Map<String, List<EntitySummary>> allProductsWithOffers =
 getAllProductsWithOffers();
```

```
 ReferenceCodesUtils.formatOutput(allProductsWithOffers);
}

public static Map<String, List<EntitySummary>> getAllProductsWithOffers() {
 MarketplaceCatalogClient marketplaceCatalogClient =
   MarketplaceCatalogClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 Map<String, List<EntitySummary>> allProductsWithOffers = new HashMap<String,
List<EntitySummary>> ();

 // get all product entities
 List<EntitySummary> productEntityList = new ArrayList<EntitySummary>();

 ListEntitiesRequest listEntitiesRequest =
   ListEntitiesRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .entityType(PRODUCT_TYPE_AMI)
   .maxResults(10)
   .nextToken(null)
   .build();


 ListEntitiesResponse listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);

 productEntityList.addAll(listEntitiesResponse.entitySummaryList());


 while (listEntitiesResponse.nextToken() != null) {
  listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(PRODUCT_TYPE_AMI)
    .maxResults(10)
    .nextToken(listEntitiesResponse.nextToken())
    .build();
  listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);
  productEntityList.addAll(listEntitiesResponse.entitySummaryList());
 }
```

```
 // loop through each product entity and get the public released offers associated
using product id filter

 for ( EntitySummary productEntitySummary : productEntityList) {
  EntityTypeFilters entityTypeFilters =
    EntityTypeFilters.builder()
    .offerFilters(OfferFilters.builder()
      .targeting(OfferTargetingFilter.builder()
        .valueListWithStrings(OFFER_TARGETING_NONE)
        .build())
      .state(OfferStateFilter.builder()
        .valueListWithStrings(OFFER_STATE_RELEASED)
        .build())
      .productId(OfferProductIdFilter.builder()
        .valueList(productEntitySummary.entityId())
        .build())
      .build())
    .build();

  listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_OFFER)
    .maxResults(10)
    .entityTypeFilters(entityTypeFilters)
    .nextToken(null)
    .build();

  listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);

  // save all entitySummary of the results into entitySummaryList

  List<EntitySummary> offerEntitySummaryList = new ArrayList<EntitySummary>();

  offerEntitySummaryList.addAll(listEntitiesResponse.entitySummaryList());

  while ( listEntitiesResponse.nextToken() != null &&
listEntitiesResponse.nextToken().length() > 0) {
   listEntitiesRequest =
     ListEntitiesRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityType(ENTITY_TYPE_OFFER)
```

```
      .maxResults(10)
      .entityTypeFilters(entityTypeFilters)
      .nextToken(listEntitiesResponse.nextToken())
      .build();
    listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);
    offerEntitySummaryList.addAll(listEntitiesResponse.entitySummaryList());
   }

   // save final results into map; key = product id; value = offer entity summary
 list

   allProductsWithOffers.put(productEntitySummary.entityId(),
 offerEntitySummaryList);
  }
  return allProductsWithOffers;
 }

}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - DescribeEntity
  - ListEntities

재판매 권한 부여

재판매 권한 부여 초안 생성

다음 코드 예제에서는 채널 파트너에게 게시하기 전에 내부적으로 검토할 수 있도록 모든 제품 유형에 대한 재판매 권한 부여 초안을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

재판매 권한 부여 설명

다음 코드 예제에서는 재판매 권한 부여를 설명하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.catalogapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
```

```
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;

public class DescribeEntity {

 /*
  * Describe my AMI or SaaS or Container product and check if it contains all the
 information I need to know about the product
  */
 public static void main(String[] args) {

  String offerId = args.length > 0 ? args[0] : OFFER_ID;

  DescribeEntityResponse describeEntityResponse =
 getDescribeEntityResponse(offerId);

  ReferenceCodesUtils.formatOutput(describeEntityResponse);
 }

 public static DescribeEntityResponse getDescribeEntityResponse(String offerId) {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeEntityRequest describeEntityRequest =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(offerId)
    .build();

  DescribeEntityResponse describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);
  return describeEntityResponse;
 }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeEntity를 참조하세요.

비공개 제안과 함께 일회성 재판매 권한 부여 게시

다음 코드 예제에서는 채널 파트너가 권한 부여를 사용하여 채널 파트너 비공개 제안(CPPO)을 생성할 수 있도록 비공개 제안과 함께 일회성 재판매 권한 부여를 게시하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 `RunChangesets`에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
```

```
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
```

```
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

만료 날짜가 있는 다중 사용 재판매 권한 부여 게시

다음 코드 예제에서는 채널 파트너가 권한 부여를 사용하여 CPPO를 생성할 수 있도록 시간당 연간 요금이 적용되는 AMI 제품에 대한 만료 날짜가 포함된 다중 사용 재판매 권한 부여를 게시하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
```

```
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
```

```
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

만료 날짜 및 EULA를 포함한 다중 사용 재판매 권한 부여 게시

다음 코드 예제에서는 모든 제품 유형에 대한 만료 날짜가 포함된 다중 사용 재판매 권한 부여를 게시하고 구매자에게 보낼 사용자 지정 EULA를 추가하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
```

```
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
```

```
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

만료 날짜 및 리셀러 계약 설명서와 함께 다중 사용 재판매 권한 부여 게시

다음 코드 예제에서는 모든 제품 유형에 대한 만료 날짜가 포함된 다중 사용 재판매 권한 부여를 게시하고 ISV와 채널 파트너 간의 리셀러 계약 설명서를 추가하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 `RunChangesets`에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    },
                    {
                        "Type": "ResaleLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomResellerContract",
                                "Url": "https://s3.amazonaws.com/aws-mp-standard-
contracts/Standard-Contact-for-AWS-Marketplace-2022-07-14.pdf"}
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
```

```
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

만료 날짜가 포함된 다중 사용 재판매 권한 부여 게시 및 특정 구매자 계정 추가

다음 코드 예제에서는 모든 제품 유형에 대한 만료 날짜가 포함된 다중 사용 재판매 권한 부여를 게시하고 재판매에 대한 특정 구매자 계정을 추가하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 `RunChangesets`에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "111111111111"
                            ]
                        }
                    }
```

```
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

만료 날짜가 없는 다중 사용 재판매 권한 부여 게시

다음 코드 예제에서는 CP가 권한 부여를 사용하여 CPPO를 생성할 수 있도록 AMI 제품에 대해 시간 단위 연간 요금이 적용되고 만료 날짜가 없는 다중 사용 재판매 권한 부여를 게시하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
```

```
                                "Value": "P12M"
                            },
                            "RateCard": [
                                {
                                    "DimensionKey": "t2.small",
                                    "Price": "150"
                                }
                            ],
                            "Constraints": {
                                "MultipleDimensionSelection": "Allowed",
                                "QuantityConfiguration": "Allowed"
                            }
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateLegalTerms",
        "Entity": {
            "Type": "ResaleAuthorization@1.0",
            "Identifier": "$ResaleAuthorization.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "BuyerLegalTerm",
                    "Documents": [
                        {
                            "Type": "CustomEula",
                            "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                        }
                    ]
                }
            ]
        }
    }
]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

만료 날짜 및 EULA가 없는 다중 사용 재판매 권한 부여 게시

다음 코드 예제에서는 모든 제품 유형에 대한 만료 날짜가 없는 다중 사용 재판매 권한 부여를 게시하고 구매자에게 보낼 사용자 지정 EULA를 추가하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 `RunChangesets`에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
```

```
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
```

```
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

만료 날짜 및 리셀러 계약 설명서 없이 다중 사용 재판매 권한 부여 게시

다음 코드 예제에서는 모든 제품 유형에 대한 만료 날짜 없이 다중 사용 재판매 권한 부여를 게시하고 ISV와 채널 파트너 간의 리셀러 계약 설명서를 추가하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
```

```
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
```

```
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    },
                    {
                        "Type": "ResaleLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomResellerContract",
                                "Url": "https://s3.amazonaws.com/aws-mp-standard-
contracts/Standard-Contact-for-AWS-Marketplace-2022-07-14.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

만료 날짜 없이 다중 사용 재판매 권한 부여 게시 및 특정 구매자 계정 추가

다음 코드 예제에서는 모든 제품 유형에 대한 만료 날짜 없이 다중 사용 재판매 권한 부여를 게시하고 재판매에 대한 특정 구매자 계정을 추가하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 `RunChangesets`에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "111111111111"
                            ]
                        }
                    }
                ]
```

```
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

일회성 재판매 권한 부여 게시 및 유연한 결제 일정 추가

다음 코드 예제에서는 모든 제품 유형에 대한 일회성 재판매 권한 부여를 게시하고 유연한 결제 일정을 추가하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleFixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.00",
                        "Duration": "P12M",
                        "Grants": [
                          {
```

```
                        "DimensionKey": "Users",
                        "MaxQuantity": 10
                    }
                ]
            }
        ]
    }
},
{
    "ChangeType": "UpdatePaymentScheduleTerms",
    "Entity": {
        "Type": "ResaleAuthorization@1.0",
        "Identifier": "$ResaleAuthorization.Entity.Identifier"
    },
    "DetailsDocument": {
        "Terms": [
            {
                "Type": "ResalePaymentScheduleTerm",
                "CurrencyCode": "USD",
                "Schedule": [
                    {
                        "ChargeDate": "2023-09-01",
                        "ChargeAmount": "200.00"
                    },
                    {
                        "ChargeDate": "2023-12-01",
                        "ChargeAmount": "250.00"
                    }
                ]
            }
        ]
    }
},
{
    "ChangeType": "UpdateAvailability",
    "Entity": {
        "Type": "ResaleAuthorization@1.0",
        "Identifier": "$ResaleAuthorization.Entity.Identifier"
    },
    "DetailsDocument": {
        "AvailabilityEndDate": "2023-06-30",
        "OffersMaxQuantity": 1
    }
},
```

```
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

일회성 재판매 권한 부여 게시 및 EULA 추가

다음 코드 예제에서는 모든 제품 유형에 대한 일회성 재판매 권한 부여를 게시하고 구매자에게 보낼 사용자 지정 EULA를 추가하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
```

```
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
```

```
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

일회성 재판매 권한 부여 게시 및 특정 구매자 계정 추가

다음 코드 예제에서는 모든 제품 유형에 대한 일회성 재판매 권한 부여를 게시하고 재판매에 대한 특정 구매자 계정을 추가하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
```

```
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": "1"
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "111111111111"
                            ]
                        }
                    }
                ]
            }
```

```
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

일회성 재판매 권한 부여 게시 및 리셀러 계약 설명서 추가

다음 코드 예제에서는 모든 제품 유형에 대한 일회성 재판매 권한 부여를 게시하고 ISV와 채널 파트너 간의 리셀러 계약 설명서를 추가하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
```

```
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
```

```
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

일회성 재판매 권한 부여 게시 및 갱신 여부 추가

다음 코드 예제에서는 모든 제품 유형에 대한 일회성 재판매 권한 부여를 게시하고 갱신 여부를 추가하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "222222222222"
                            ]
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType":"UpdateInformation",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "PreExistingBuyerAgreement": {
                    "AcquisitionChannel": "AwsMarketplace",
                    "PricingModel": "Contract"
                }
             }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

재판매 권한 부여 제한

다음 코드 예제에서는 재판매 권한 부여를 제한하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
```

```
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "RestrictResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "resaleauthz-1111111111111"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

일회성 또는 다중 사용 재판매 권한 부여의 이름 및 설명 업데이트

다음 코드 예제에서는 모든 제품 유형에 대해 권한 부여를 게시하기 전에 일회성 또는 다중 사용 재판매 권한 부여의 이름과 설명을 업데이트하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType":"UpdateInformation",
            "Entity": {
             "Type": "ResaleAuthorization@1.0",
             "Identifier": "resaleauthz-1111111111111"
            },
            "DetailsDocument": {
             "Name": "TestResaleAuthorization",
```

```
                "Description": "Worldwide ResaleAuthorization for Test Product"
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

SaaS 제품

공개 제안 초안을 사용하여 SaaS 제품 초안 생성

다음 코드 예제에서는 공개 제안 초안을 사용하여 SaaS 제품 초안을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "DetailsDocument": {
              "ProductTitle": "Sample product"
            }
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
```

```
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier",
                "Name": "Test Offer"
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

공개 또는 제한된 SaaS 제품 및 계약 요금이 적용되는 공개 제안 생성

다음 코드 예제에서는 공개 또는 제한된 SaaS 제품 및 계약 요금이 적용되는 공개 제안을 생성하는 방법을 보여줍니다. 이 예제에서는 표준 또는 사용자 지정 EULA를 생성합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 `RunChangesets`에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
```

```
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
```

```
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P1M"
```

```
                            },
                            "RateCard": [
                                {
                                    "DimensionKey": "BasicService",
                                    "Price": "20"
                                },
                                {
                                    "DimensionKey": "PremiumService",
                                    "Price": "25"
                                }
                            ],
                            "Constraints": {
                                "MultipleDimensionSelection": "Allowed",
                                "QuantityConfiguration": "Allowed"
                            }
                        },
                        {
                            "Selector": {
                                "Type": "Duration",
                                "Value": "P12M"
                            },
                            "RateCard": [
                                {
                                    "DimensionKey": "BasicService",
                                    "Price": "150"
                                },
                                {
                                    "DimensionKey": "PremiumService",
                                    "Price": "300"
                                }
                            ],
                            "Constraints": {
                                "MultipleDimensionSelection": "Allowed",
                                "QuantityConfiguration": "Allowed"
                            }
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateLegalTerms",
        "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

공개 또는 제한된 SaaS 제품 및 종량제 요금으로 계약이 적용되는 공개 제안 생성

다음 코드 예제에서는 공개 또는 제한된 SaaS 제품 및 종량제 요금으로 계약이 적용되는 공개 제안을 생성하는 방법을 보여줍니다. 이 예제에서는 표준 또는 사용자 지정 EULA를 생성합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
```

```
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
                            }
                        }
                    }
                ]
            }
```

```
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "WorkloadSmall",
                    "Description": "Workload: Per medium instance",
                    "Name": "Workload: Per medium instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "WorkloadMedium",
                    "Description": "Workload: Per large instance",
                    "Name": "Workload: Per large instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                }
```

```
            ]
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
```

```
                    "CurrencyCode": "USD",
                    "RateCards": [
                        {
                            "RateCard": [
                                {
                                    "DimensionKey": "WorkloadSmall",
                                    "Price": "0.15"
                                },
                                {
                                    "DimensionKey": "WorkloadMedium",
                                    "Price": "0.25"
                                }
                            ]
                        }
                    ]
                },
                {
                    "Type": "ConfigurableUpfrontPricingTerm",
                    "CurrencyCode": "USD",
                    "RateCards": [
                        {
                            "Selector": {
                                "Type": "Duration",
                                "Value": "P12M"
                            },
                            "RateCard": [
                                {
                                    "DimensionKey": "BasicService",
                                    "Price": "150"
                                },
                                {
                                    "DimensionKey": "PremiumService",
                                    "Price": "300"
                                }
                            ],
                            "Constraints": {
                                "MultipleDimensionSelection": "Allowed",
                                "QuantityConfiguration": "Allowed"
                            }
                        }
                    ]
                }
            ]
        }
```

```
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
```

```
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

공개 또는 제한된 SaaS 제품 및 구독 요금이 적용되는 공개 제안 생성

다음 코드 예제에서는 구독 요금이 적용되는 공개 또는 제한된 SaaS 제품 및 공개 제안을 생성하는 방법을 보여줍니다. 이 예제에서는 표준 또는 사용자 지정 EULA를 생성합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 `RunChangesets`에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
```

```
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
```

```
                        }
                    }
                }
            ]
        }
    },
    {
        "ChangeType": "AddDimensions",
        "Entity": {
            "Type": "SaaSProduct@1.0",
            "Identifier": "$CreateProductChange.Entity.Identifier"
        },
        "DetailsDocument": [
            {
                "Key": "WorkloadSmall",
                "Description": "Workload: Per medium instance",
                "Name": "Workload: Per medium instance",
                "Types": [
                    "ExternallyMetered"
                ],
                "Unit": "Units"
            },
            {
                "Key": "WorkloadMedium",
                "Description": "Workload: Per large instance",
                "Name": "Workload: Per large instance",
                "Types": [
                    "ExternallyMetered"
                ],
                "Unit": "Units"
            }
        ]
    },
    {
        "ChangeType": "ReleaseProduct",
        "Entity": {
            "Type": "SaaSProduct@1.0",
            "Identifier": "$CreateProductChange.Entity.Identifier"
        },
        "DetailsDocument": {}
    },
    {
        "ChangeType": "CreateOffer",
        "Entity": {
```

```
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
```

```
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateLegalTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "LegalTerm",
                    "Documents": [
                        {
                            "Type": "StandardEula",
                            "Version": "2022-07-14"
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateSupportTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "SupportTerm",
                    "RefundPolicy": "Absolutely no refund, period."
                }
            ]
        }
    },
    {
        "ChangeType": "ReleaseOffer",
        "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

SaaS 제품 및 관련된 공개 제안 게시

다음 코드 예제에서는 SaaS 제품 및 관련된 공개 제안을 게시하는 방법을 보여줍니다. 제품은 기본적으로 제한된 상태입니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 RunChangesets에 전달합니다.

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://bucketname.s3.amazonaws.com/logo.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
```

```
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl": "https://www.aws.amazon.com/
marketplace/management"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "New Test Offer",
                "Description": "New offer description"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Updated refund policy description"
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P1M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "20"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "25"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            },
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
```

```
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateRenewalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "RenewalTerm"
                    }
                ]
            }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

기존 초안을 바탕으로 SaaS 제품 및 관련된 공개 제안 게시

다음 코드 예제에서는 기존 초안을 바탕으로 SaaS 제품 및 관련된 공개 제안을 게시하는 방법을 보여줍니다. 제품은 기본적으로 제한된 상태입니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateVisibility",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "TargetVisibility": "Public"
            }
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

AMI 또는 SaaS 제품의 차원 업데이트

다음 코드 예제에서는 AMI 또는 SaaS 제품의 차원을 업데이트하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

이 예제를 실행하려면 유틸리티 섹션의 changeset를 시작하기 위한 유틸리티에서 다음 JSON changeset를 `RunChangesets`에 전달합니다.

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "prod-111111111111"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Types": [
                        "Entitled"
                    ],
                    "Name": "Some new name",
                    "Description": "Some new description"
                }
            ]
        }
    ]
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

유틸리티

changeset를 시작하기 위한 유틸리티

다음 코드 예제에서는 유틸리티를 정의하여 changeset를 시작하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

JSON 파일에서 changeset를 로드하고 처리를 시작하는 유틸리티입니다.

```
package com.example.awsmarketplace.catalogapi;

import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

import org.apache.commons.io.IOUtils;
import org.apache.commons.lang3.StringUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.protocols.json.internal.unmarshall.document.DocumentUnmarshaller;
import software.amazon.awssdk.protocols.jsoncore.JsonNodeParser;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import software.amazon.awssdk.services.marketplacecatalog.model.Change;
import software.amazon.awssdk.services.marketplacecatalog.model.Entity;
import
 software.amazon.awssdk.services.marketplacecatalog.model.StartChangeSetRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.StartChangeSetResponse;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.ToNumberPolicy;
```

```
import com.example.awsmarketplace.catalogapi.Entity.ChangeSet;
import com.example.awsmarketplace.catalogapi.Entity.ChangeSetEntity;
import com.example.awsmarketplace.catalogapi.Entity.Root;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;
import com.example.awsmarketplace.utils.StringSerializer;

/**
 * Before running this Java V2 code example, convert all Details attribute to
 DetailsDocument if any
 */

public class RunChangesets {

 private static final Gson GSON = new GsonBuilder()
   .setObjectToNumberStrategy(ToNumberPolicy.LAZILY_PARSED_NUMBER)
   .registerTypeAdapter(String.class, new StringSerializer())
   .create();

 public static void main(String[] args) {

  // input json can be specified here or passed from input parameter
  String inputChangeSetFile = "changeSets/offers/
CreateReplacementOfferFromAGWithContractPricingDetailDocument.json";

  if (args.length > 0)
   inputChangeSetFile = args[0];

  // parse the input changeset file to string for process
  String changeSetsInput = readChangeSetToString(inputChangeSetFile);

  // process the changeset request
  try {
   StartChangeSetResponse result = getChangeSetRequestResult(changeSetsInput);
   ReferenceCodesUtils.formatOutput(result);
  } catch (Exception e) {
   e.printStackTrace();
  }
 }

 public static StartChangeSetResponse getChangeSetRequestResult(String
 changeSetsInput) throws IOException {

  //set up AWS credentials
  MarketplaceCatalogClient marketplaceCatalogClient =
```

```
  MarketplaceCatalogClient.builder()
  .httpClient(ApacheHttpClient.builder().build())
  .credentialsProvider(ProfileCredentialsProvider.create())
  .build();

//changeset list to save all the changesets in the changesets file
List<Change> changeSetLists = new ArrayList<Change>();

// read all changesets into object
Root root = GSON.fromJson(changeSetsInput, Root.class);

// process each changeset and add each changeset request to changesets list
for (ChangeSet cs : root.changeSet) {

 ChangeSetEntity entity = cs.Entity;
 String entityType = entity.Type;
 String entityIdentifier = StringUtils.defaultIfBlank(entity.Identifier, null);
 Document detailsDocument = getDocumentFromObject(cs.DetailsDocument);

 Entity awsEntity =
   Entity.builder()
   .type(entityType)
   .identifier(entityIdentifier)
   .build();

 Change inputChangeRequest =
   Change.builder()
   .changeType(cs.ChangeType)
   .changeName(cs.ChangeName)
   .entity(awsEntity)
   .detailsDocument(detailsDocument)
   .build();

 changeSetLists.add(inputChangeRequest);
}

// process all changeset requests
StartChangeSetRequest startChangeSetRequest =
  StartChangeSetRequest.builder()
  .catalog(root.catalog)
  .changeSet(changeSetLists)
  .build();
```

```
 StartChangeSetResponse result =
marketplaceCatalogClient.startChangeSet(startChangeSetRequest);

 return result;
}

public static Document getDocumentFromObject(Object detailsObject) {

 String detailsString = "{}";
 try {
  detailsString = IOUtils.toString(new
ByteArrayInputStream(GSON.toJson(detailsObject).getBytes()), "UTF-8");
 } catch (IOException e) {
  e.printStackTrace();
 }

 JsonNodeParser jsonNodeParser = JsonNodeParser.create();
 Document doc = jsonNodeParser.parse(detailsString).visit(new
DocumentUnmarshaller());
 return doc;
}


public static String readChangeSetToString (String inputChangeSetFile) {

 InputStream changesetInputStream =
RunChangesets.class.getClassLoader().getResourceAsStream(inputChangeSetFile);

 String changeSetsInput = null;

 try {
  changeSetsInput = IOUtils.toString(changesetInputStream, "UTF-8");
 } catch (IOException e) {
  e.printStackTrace();
 }

 return changeSetsInput;

}
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartChangeSet를 참조하세요.

AWS Marketplace SDK for Java 2.x를 사용한 계약 API 예제

다음 코드 예제에서는 AWS Marketplace 계약 API와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

계약

모든 계약 ID 가져오기

다음 코드 예제에서는 모든 계약 ID를 가져오는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;
```

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAllAgreementsIds {

 /*
  * Get all purchase agreements ids with party type = proposer;
  * Depend on the number of agreements in your account, this code may take some time
 to finish.
  */
 public static void main(String[] args) {

  List<String> agreementIds = getAllAgreementIds();

  ReferenceCodesUtils.formatOutput(agreementIds);

 }

 public static List<String> getAllAgreementIds() {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  // get all filters
  Filter partyType = Filter.builder().name(PARTY_TYPE_FILTER_NAME)
    .values(PARTY_TYPE_FILTER_VALUE_PROPOSER).build();

  Filter agreementType = Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME)
    .values(AGREEMENT_TYPE_FILTER_VALUE_PURCHASEAGREEMENT).build();

  List<Filter> searchFilters = new ArrayList<Filter>();

  searchFilters.addAll(Arrays.asList(partyType, agreementType));

  // Save all results in a list array
  List<AgreementViewSummary> agreementSummaryList = new
 ArrayList<AgreementViewSummary>();
```

```
  SearchAgreementsRequest searchAgreementsRequest =
    SearchAgreementsRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .filters(searchFilters)
    .build();

  SearchAgreementsResponse searchAgreementsResponse =
 marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

  agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());

  while (searchAgreementsResponse.nextToken() != null &&
 searchAgreementsResponse.nextToken().length() > 0) {
   searchAgreementsRequest =
     SearchAgreementsRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .nextToken(searchAgreementsResponse.nextToken())
     .filters(searchFilters)
     .build();
   searchAgreementsResponse =
 marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
   agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());
  }

  List<String> agreementIds = new ArrayList<String>();
  for (AgreementViewSummary summary : agreementSummaryList) {
   agreementIds.add(summary.agreementId());
  }
  return agreementIds;
 }

}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 SearchAgreements를 참조하세요.

모든 계약 가져오기

다음 코드 예제에서는 모든 계약을 가져오는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAllAgreements {

 /*
  * Get all purchase agreements with party type = proposer;
  * Depend on the number of agreements in your account, this code may take some time
 to finish.
  */
 public static void main(String[] args) {

  List<AgreementViewSummary> agreementSummaryList = getAllAgreements();
```

```
 ReferenceCodesUtils.formatOutput(agreementSummaryList);
}

public static List<AgreementViewSummary> getAllAgreements() {
 MarketplaceAgreementClient marketplaceAgreementClient =
   MarketplaceAgreementClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 // get all filters

 Filter partyType = Filter.builder().name(PARTY_TYPE_FILTER_NAME)
   .values(PARTY_TYPE_FILTER_VALUE_PROPOSER).build();

 Filter agreementType = Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME)
   .values(AGREEMENT_TYPE_FILTER_VALUE_PURCHASEAGREEMENT).build();

 List<Filter> searchFilters = new ArrayList<Filter>();

 searchFilters.addAll(Arrays.asList(partyType, agreementType));

 // Save all results in a list array

 List<AgreementViewSummary> agreementSummaryList = new
ArrayList<AgreementViewSummary>();

 SearchAgreementsRequest searchAgreementsRequest =
   SearchAgreementsRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .filters(searchFilters)
   .build();

 SearchAgreementsResponse searchAgreementsResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

 agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());

 while (searchAgreementsResponse.nextToken() != null &&
searchAgreementsResponse.nextToken().length() > 0) {
  searchAgreementsRequest =
    SearchAgreementsRequest.builder()
    .catalog(AWS_MP_CATALOG)
```

```
            .nextToken(searchAgreementsResponse.nextToken())
            .filters(searchFilters).build();
        searchAgreementsResponse =
     marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
        agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());
      }
      return agreementSummaryList;
     }

    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 SearchAgreements를 참조하세요.

계약에서 고객 ID 가져오기

다음 코드 예제에서는 계약에서 고객 ID를 가져오는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;

public class GetAgreementCustomerInfo {
```

```
 /*
  * Obtain metadata about the customer who created the agreement, such as the
 customer's AWS Account ID
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  DescribeAgreementResponse describeAgreementResponse =
 getDescribeAgreementResponse(agreementId);

  System.out.println("Customer's AWS Account ID is " +
 describeAgreementResponse.acceptor().accountId());

 }

 public static DescribeAgreementResponse getDescribeAgreementResponse(String
 agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeAgreementRequest describeAgreementRequest =
    DescribeAgreementRequest.builder()
    .agreementId(agreementId)
    .build();

  DescribeAgreementResponse describeAgreementResponse =
 marketplaceAgreementClient.describeAgreement(describeAgreementRequest);
  return describeAgreementResponse;
 }

}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeAgreement를 참조하세요.

계약에서 재무 세부 정보 가져오기

다음 코드 예제에서는 계약에서 재무 세부 정보를 가져오는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;

public class GetAgreementFinancialDetails {

 /*
  * Obtain financial details, such as Total Contract Value of the agreement from a
 given agreement
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  String totalContractValue = getTotalContractValue(agreementId);

  System.out.println("Total Contract Value is " + totalContractValue);

 }

 public static String getTotalContractValue(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
```

```
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeAgreementRequest describeAgreementRequest =
    DescribeAgreementRequest.builder()
    .agreementId(agreementId)
    .build();

  DescribeAgreementResponse describeAgreementResponse =
 marketplaceAgreementClient.describeAgreement(describeAgreementRequest);

  String totalContractValue = "N/A";

  if ( describeAgreementResponse.estimatedCharges() != null ) {
   totalContractValue =
 describeAgreementResponse.estimatedCharges().agreementValue()
     + " "
     + describeAgreementResponse.estimatedCharges().currencyCode();
  }
  return totalContractValue;
 }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeAgreement를 참조하세요.

계약에서 무료 평가판 세부 정보 가져오기

다음 코드 예제에서는 계약에서 무료 평가판 세부 정보를 가져오는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;
```

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.FreeTrialPricingTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;

import java.util.ArrayList;
import java.util.List;

import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsFreeTrialDetails {

 /*
  * Obtain the details from an agreement of a free trial I have provided to the
 customer
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<FreeTrialPricingTerm> freeTrialPricingTerms =
 getFreeTrialPricingTerms(agreementId);

  ReferenceCodesUtils.formatOutput(freeTrialPricingTerms);
 }

 public static List<FreeTrialPricingTerm> getFreeTrialPricingTerms(String
 agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();
```

```
  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
     .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  List<FreeTrialPricingTerm> freeTrialPricingTerms = new
 ArrayList<FreeTrialPricingTerm>();

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   if (acceptedTerm.freeTrialPricingTerm() != null) {
    freeTrialPricingTerms.add(acceptedTerm.freeTrialPricingTerm());
   }
  }
  return freeTrialPricingTerms;
 }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeAgreement를 참조하세요.

계약 정보 가져오기

다음 코드 예제에서는 계약에 대한 정보를 가져오는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;
```

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;

public class DescribeAgreement {

 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  DescribeAgreementResponse describeAgreementResponse = getResponse(agreementId);

  ReferenceCodesUtils.formatOutput(describeAgreementResponse);

 }

 public static DescribeAgreementResponse getResponse(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeAgreementRequest describeAgreementRequest =
    DescribeAgreementRequest.builder()
    .agreementId(agreementId)
    .build();

  DescribeAgreementResponse describeAgreementResponse =
 marketplaceAgreementClient.describeAgreement(describeAgreementRequest);
  return describeAgreementResponse;
 }

}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeAgreement를 참조하세요.

계약에서 제품 및 제안 세부 정보 가져오기

다음 코드 예제에서는 계약에서 제품 및 제안 세부 정보를 가져오는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.Resource;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;

public class GetProductAndOfferDetailFromAgreement {

 public static void main(String[] args) {

  // call Agreement API to get offer and product information for the agreement
```

```
 String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

 List<DescribeEntityResponse> entityResponseList = getEntities(agreementId);

 for (DescribeEntityResponse response : entityResponseList) {
  ReferenceCodesUtils.formatOutput(response);
 }
}

public static List<DescribeEntityResponse> getEntities(String agreementId) {
 List<DescribeEntityResponse> entityResponseList = new
ArrayList<DescribeEntityResponse> ();

 MarketplaceAgreementClient marketplaceAgreementClient =
   MarketplaceAgreementClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 DescribeAgreementRequest describeAgreementRequest =
   DescribeAgreementRequest.builder()
   .agreementId(agreementId)
   .build();

 DescribeAgreementResponse describeAgreementResponse =
marketplaceAgreementClient.describeAgreement(describeAgreementRequest);

 // get offer id for the given agreement

 String offerId = describeAgreementResponse.proposalSummary().offerId();

 // get all the product ids for this agreement

 List<String> productIds = new ArrayList<String>();
 for (Resource resource : describeAgreementResponse.proposalSummary().resources())
{
  productIds.add(resource.id());
 }

 // call Catalog API to get the details of the offer and products

 MarketplaceCatalogClient marketplaceCatalogClient =
   MarketplaceCatalogClient.builder()
```

```
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeEntityRequest describeEntityRequest =
    DescribeEntityRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityId(offerId).build();

  DescribeEntityResponse describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);

  entityResponseList.add(describeEntityResponse);

  for (String productId : productIds) {
   describeEntityRequest =
     DescribeEntityRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityId(productId).build();
   describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);
   System.out.println("Print details for product " + productId);
   entityResponseList.add(describeEntityResponse);
  }
  return entityResponseList;
 }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeAgreement를 참조하세요.

계약의 EULA 가져오기

다음 코드 예제에서는 계약의 EULA를 가져오는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.DocumentItem;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsEula {

 /*
  * Obtain the EULA I have entered into with my customer via the agreement
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<DocumentItem> legalEulaArray = getLegalEula(agreementId);

  ReferenceCodesUtils.formatOutput(legalEulaArray);
 }

 public static List<DocumentItem> getLegalEula(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
```

```
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  List<DocumentItem> legalEulaArray = new ArrayList<>();

  getAgreementTermsResponse.acceptedTerms().stream()
      .filter(acceptedTerm -> acceptedTerm.legalTerm() != null &&
 acceptedTerm.legalTerm().hasDocuments())
      .flatMap(acceptedTerm -> acceptedTerm.legalTerm().documents().stream())
      .filter(docItem -> docItem.type() != null)
      .forEach(legalEulaArray::add);
  return legalEulaArray;
 }

}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetAgreementTerms을 참조하세요.

계약의 자동 갱신 조건 가져오기

다음 코드 예제에서는 계약의 자동 갱신 조건을 가져오는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

public class GetAgreementAutoRenewal {

 /*
  * Obtain the auto-renewal status of the agreement
  */

 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  String autoRenewal = getAutoRenewal(agreementId);

  System.out.println("Auto-Renewal status is " + autoRenewal);
 }

 public static String getAutoRenewal(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder()
    .agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  String autoRenewal = "No Auto Renewal";

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   if (acceptedTerm.renewalTerm() != null &&
 acceptedTerm.renewalTerm().configuration() != null
     && acceptedTerm.renewalTerm().configuration().enableAutoRenew() != null) {
```

```
    autoRenewal =
 String.valueOf(acceptedTerm.renewalTerm().configuration().enableAutoRenew().booleanValue())
    break;
   }
  }
  return autoRenewal;
 }

}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetAgreementTerms을 참조하세요.

계약에서 구매한 차원 가져오기

다음 코드 예제에서는 계약에서 구매한 차원을 가져오는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

import java.util.ArrayList;
import java.util.List;
```

```
import static com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsDimensionPurchased {

 /*
  * Obtain the dimensions the buyer has purchased from me via the agreement
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<String> dimensionKeys = getDimensionKeys(agreementId);

  ReferenceCodesUtils.formatOutput(dimensionKeys);
 }

 public static List<String> getDimensionKeys(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  List<String> dimensionKeys = new ArrayList<String>();
  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   if (acceptedTerm.configurableUpfrontPricingTerm() != null) {
    if
 (acceptedTerm.configurableUpfrontPricingTerm().configuration().selectorValue() !=
 null) {
     List<Dimension> dimensions =
 acceptedTerm.configurableUpfrontPricingTerm().configuration().dimensions();
     for (Dimension dimension : dimensions) {
      dimensionKeys.add(dimension.dimensionKey());
     }
    }
```

```
   }
  }
  return dimensionKeys;
 }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetAgreementTerms을 참조하세요.

계약에서 구매한 각 차원의 인스턴스 가져오기

다음 코드 예제에서는 계약에서 구매한 각 차원의 인스턴스를 가져오는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import software.amazon.awssdk.services.marketplaceagreement.model.Dimension;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
```

```
import static com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsDimensionInstances {

 /*
  * get instances of each dimension that buyer has purchased in the agreement
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  Map<String, List<Dimension>> dimensionMap = getDimensions(agreementId);

  ReferenceCodesUtils.formatOutput(dimensionMap);
 }

 public static Map<String, List<Dimension>> getDimensions(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  Map<String, List<Dimension>> dimensionMap = new HashMap<String,
 List<Dimension>>();

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   List<Dimension> dimensionsList = new ArrayList<Dimension>();
   if (acceptedTerm.configurableUpfrontPricingTerm() != null) {
    String selectorValue = "";
    if (acceptedTerm.configurableUpfrontPricingTerm().configuration() != null) {
     if
 (acceptedTerm.configurableUpfrontPricingTerm().configuration().selectorValue() !=
 null) {
      selectorValue =
 acceptedTerm.configurableUpfrontPricingTerm().configuration().selectorValue();
```

```
        }
        if
 (acceptedTerm.configurableUpfrontPricingTerm().configuration().hasDimensions()) {
         dimensionsList =
 acceptedTerm.configurableUpfrontPricingTerm().configuration().dimensions();
        }
       }
       if (selectorValue.length() > 0) {
        dimensionMap.put(selectorValue, dimensionsList);
       }
      }
     }
     return dimensionMap;
 }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetAgreementTerms을 참조하세요.

계약의 결제 일정 가져오기

다음 코드 예제에서는 계약의 결제 일정을 가져오는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
```

```
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.PaymentScheduleTerm;
import software.amazon.awssdk.services.marketplaceagreement.model.ScheduleItem;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsPaymentSchedule {

 /*
  * Obtain the payment schedule I have agreed to with the agreement, including the
 invoice date and invoice amount
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<Map<String, Object>> paymentScheduleArray = getPaymentSchedules(agreementId);

  ReferenceCodesUtils.formatOutput(paymentScheduleArray);
 }

 public static List<Map<String, Object>> getPaymentSchedules(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);
  List<Map<String, Object>> paymentScheduleArray = new ArrayList<>();
```

```
  String currencyCode = "";

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   if (acceptedTerm.paymentScheduleTerm() != null) {
    PaymentScheduleTerm paymentScheduleTerm = acceptedTerm.paymentScheduleTerm();
    if (paymentScheduleTerm.currencyCode() != null) {
     currencyCode = paymentScheduleTerm.currencyCode();
    }
    if (paymentScheduleTerm.hasSchedule()) {
     for (ScheduleItem schedule : paymentScheduleTerm.schedule()) {
      if (schedule.chargeDate() != null) {
       String chargeDate = schedule.chargeDate().toString();
       String chargeAmount = schedule.chargeAmount();
       Map<String, Object> scheduleMap = new HashMap<>();
       scheduleMap.put(ATTRIBUTE_CURRENCY_CODE, currencyCode);
       scheduleMap.put(ATTRIBUTE_CHARGE_DATE, chargeDate);
       scheduleMap.put(ATTRIBUTE_CHARGE_AMOUNT, chargeAmount);
       paymentScheduleArray.add(scheduleMap);
      }
     }
    }
   }
  }
  return paymentScheduleArray;
 }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetAgreementTerms을 참조하세요.

계약에서 차원당 요금 가져오기

다음 코드 예제는 계약에서 차원당 요금을 가져오는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsPricingEachDimension {

 /*
  * Obtain pricing per each dimension in the agreement
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<Object> dimensions = getDimensions(agreementId);

  ReferenceCodesUtils.formatOutput(dimensions);
 }

 public static List<Object> getDimensions(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
```

```
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  List<Object> dimensions = new ArrayList<Object>();

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   List<Object> rateInfo = new ArrayList<Object>();
   if (acceptedTerm.configurableUpfrontPricingTerm() != null) {
    if (acceptedTerm.configurableUpfrontPricingTerm().type() != null) {
     rateInfo.add(acceptedTerm.configurableUpfrontPricingTerm().type());
    }
    if (acceptedTerm.configurableUpfrontPricingTerm().currencyCode() != null) {
     rateInfo.add(acceptedTerm.configurableUpfrontPricingTerm().currencyCode());
    }
    if (acceptedTerm.configurableUpfrontPricingTerm().hasRateCards()) {
     rateInfo.add(acceptedTerm.configurableUpfrontPricingTerm().rateCards());
    }
    dimensions.add(rateInfo);
   }
  }
  return dimensions;
 }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetAgreementTerms을 참조하세요.

계약의 요금 유형 가져오기

다음 코드 예제에서는 계약의 요금 유형을 가져오는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import com.fasterxml.jackson.annotation.JsonAutoDetect.Visibility;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.Objects;
import java.util.Set;

import org.apache.commons.lang3.tuple.Triple;

import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.fasterxml.jackson.annotation.PropertyAccessor;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonNode;
```

```
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.ObjectWriter;
import com.fasterxml.jackson.datatype.jsr310.JavaTimeModule;

/*
 * Obtain the pricing type of the agreement (contract, FPS, metered, free etc.)
 */
public class GetAgreementPricingType {

 private static final String FILTER_NAME = "OfferId";

 private static final String FILTER_VALUE = OFFER_ID;

 // Product types
 private static final String SAAS_PRODUCT = "SaaSProduct";
 private static final String AMI_PRODUCT = "AmiProduct";
 private static final String ML_PRODUCT = "MachineLearningProduct";
 private static final String CONTAINER_PRODUCT = "ContainerProduct";
 private static final String DATA_PRODUCT = "DataProduct";
 private static final String PROSERVICE_PRODUCT = "ProfessionalServicesProduct";
 private static final String AIQ_PRODUCT = "AiqProduct";

 // Pricing types
 private static final String CCP = "CCP";
 private static final String ANNUAL = "Annual";
 private static final String CONTRACT = "Contract";
 private static final String SFT = "SaaS Free Trial";
 private static final String HMA = "Hourly and Monthly Agreements";
 private static final String HOURLY = "Hourly";
 private static final String MONTHLY = "Monthly";
 private static final String AFPS = "Annual FPS";
 private static final String CFPS = "Contract FPS";
 private static final String CCPFPS = "CCP with FPS";
 private static final String BYOL = "BYOL";
 private static final String FREE = "Free";
 private static final String FTH = "Free Trials and Hourly";

 // Agreement term pricing types
 private static final Set<String> LEGAL = Set.of("LegalTerm");
 private static final Set<String> CONFIGURABLE_UPFRONT =
 Set.of("ConfigurableUpfrontPricingTerm");
 private static final Set<String> USAGE_BASED = Set.of("UsageBasedPricingTerm");
 private static final Set<String> CONFIGURABLE_UPFRONT_AND_USAGE_BASED =
 Set.of("ConfigurableUpfrontPricingTerm", "UsageBasedPricingTerm");
```

```
private static final Set<String> FREE_TRIAL = Set.of("FreeTrialPricingTerm");
private static final Set<String> RECURRING_PAYMENT =
Set.of("RecurringPaymentTerm");
private static final Set<String> USAGE_BASED_AND_RECURRING_PAYMENT =
Set.of("UsageBasedPricingTerm", "RecurringPaymentTerm");
private static final Set<String> FIXED_UPFRONT_AND_PAYMENT_SCHEDULE =
Set.of("FixedUpfrontPricingTerm", "PaymentScheduleTerm");
private static final Set<String> FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED
= Set.of("FixedUpfrontPricingTerm", "PaymentScheduleTerm",
"UsageBasedPricingTerm");
private static final Set<String> BYOL_PRICING = Set.of("ByolPricingTerm");
private static final Set<String> FREE_TRIAL_AND_USAGE_BASED =
Set.of("FreeTrialPricingTerm", "UsageBasedPricingTerm");

private static final List<Set<String>> ALL_AGREEMENT_TERM_TYPES_COMBINATION
= Arrays.asList(LEGAL, CONFIGURABLE_UPFRONT, USAGE_BASED,
CONFIGURABLE_UPFRONT_AND_USAGE_BASED,
  FREE_TRIAL, RECURRING_PAYMENT, USAGE_BASED_AND_RECURRING_PAYMENT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, BYOL_PRICING,
FREE_TRIAL_AND_USAGE_BASED);

private static  MarketplaceAgreementClient marketplaceAgreementClient =
  MarketplaceAgreementClient.builder()
  .httpClient(ApacheHttpClient.builder().build())
  .credentialsProvider(ProfileCredentialsProvider.create())
  .build();

private static MarketplaceCatalogClient marketplaceCatalogClient =
  MarketplaceCatalogClient.builder()
  .httpClient(ApacheHttpClient.builder().build())
  .credentialsProvider(ProfileCredentialsProvider.create())
  .build();

   /*
    * Get agreement Pricing Type given product type, agreement term types and offer
types if needed
    */
public static String getPricingType(String productType, Set<String>
agreementTermType, Set<String> offerType) {
 Map<Triple<String, Set<String>, Set<String>>, String> pricingTypes = new
HashMap<>();
```

```
 pricingTypes.put(Triple.of(SAAS_PRODUCT, CONFIGURABLE_UPFRONT_AND_USAGE_BASED, new
HashSet<>()), CCP);
 pricingTypes.put(Triple.of(DATA_PRODUCT, CONFIGURABLE_UPFRONT_AND_USAGE_BASED, new
HashSet<>()), CCP);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT_AND_USAGE_BASED), ANNUAL);
 pricingTypes.put(Triple.of(AMI_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT_AND_USAGE_BASED), ANNUAL);
 pricingTypes.put(Triple.of(ML_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT_AND_USAGE_BASED), ANNUAL);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT), CONTRACT);
 pricingTypes.put(Triple.of(AMI_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT), CONTRACT);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, CONFIGURABLE_UPFRONT, new HashSet<>()),
CONTRACT);
 pricingTypes.put(Triple.of(DATA_PRODUCT, CONFIGURABLE_UPFRONT, new HashSet<>()),
CONTRACT);
 pricingTypes.put(Triple.of(AIQ_PRODUCT, CONFIGURABLE_UPFRONT, new HashSet<>()),
CONTRACT);
 pricingTypes.put(Triple.of(PROSERVICE_PRODUCT, CONFIGURABLE_UPFRONT, new
HashSet<>()), CONTRACT);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, FREE_TRIAL, new HashSet<>()), SFT);
 pricingTypes.put(Triple.of(AMI_PRODUCT, USAGE_BASED_AND_RECURRING_PAYMENT, new
HashSet<>()), HMA);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, USAGE_BASED, new HashSet<>()), HOURLY);
 pricingTypes.put(Triple.of(AMI_PRODUCT, USAGE_BASED, new HashSet<>()), HOURLY);
 pricingTypes.put(Triple.of(ML_PRODUCT, USAGE_BASED, new HashSet<>()), HOURLY);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, RECURRING_PAYMENT, new HashSet<>()),
MONTHLY);
 pricingTypes.put(Triple.of(AMI_PRODUCT, RECURRING_PAYMENT, new HashSet<>()),
MONTHLY);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED), AFPS);
 pricingTypes.put(Triple.of(AMI_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED), AFPS);
 pricingTypes.put(Triple.of(ML_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE, new
HashSet<>()), AFPS);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
new HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(AMI_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE), CFPS);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE, new
HashSet<>()), CFPS);
```

```
 pricingTypes.put(Triple.of(DATA_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE, new
HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(AIQ_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE, new
HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(PROSERVICE_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
new HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(SAAS_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, new HashSet<>()), CCPFPS);
 pricingTypes.put(Triple.of(DATA_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, new HashSet<>()), CCPFPS);
 pricingTypes.put(Triple.of(AIQ_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, new HashSet<>()), CCPFPS);
 pricingTypes.put(Triple.of(PROSERVICE_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, new HashSet<>()), CCPFPS);
 pricingTypes.put(Triple.of(AMI_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(PROSERVICE_PRODUCT, BYOL_PRICING, new HashSet<>()),
BYOL);
 pricingTypes.put(Triple.of(AIQ_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(ML_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, BYOL_PRICING, new HashSet<>()),
BYOL);
 pricingTypes.put(Triple.of(DATA_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, LEGAL, new HashSet<>()), FREE);
 pricingTypes.put(Triple.of(AMI_PRODUCT, FREE_TRIAL_AND_USAGE_BASED, new
HashSet<>()), FTH);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, FREE_TRIAL_AND_USAGE_BASED, new
HashSet<>()), FTH);
 pricingTypes.put(Triple.of(ML_PRODUCT, FREE_TRIAL_AND_USAGE_BASED, new
HashSet<>()), FTH);

 Triple<String, Set<String>, Set<String>> key = Triple.of(productType,
agreementTermType, offerType);

 if (pricingTypes.containsKey(key)) {
  return pricingTypes.get(key);
 } else {
  return "Unknown";
 }
}

/*
 * Given product type and agreement term types, some combinations need to check
offer term types as well.
```

```
 */
public static String needToCheckOfferTermsType(String productType, Set<String>
agreementTermTypes) {
 Map<KeyPair, String> offerTermTypes = new HashMap<>();
 offerTermTypes.put(new KeyPair(CONTAINER_PRODUCT, CONFIGURABLE_UPFRONT), "Y");
 offerTermTypes.put(new KeyPair(AMI_PRODUCT, CONFIGURABLE_UPFRONT), "Y");
 offerTermTypes.put(new KeyPair(CONTAINER_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE), "Y");
 offerTermTypes.put(new KeyPair(AMI_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE),
"Y");

 KeyPair key = new KeyPair(productType, agreementTermTypes);
 if (offerTermTypes.containsKey(key)) {
  return offerTermTypes.get(key);
 } else {
  return null;
 }
}

public static List<AgreementViewSummary> getAgreementsById() {

 List<AgreementViewSummary> agreementSummaryList = new
ArrayList<AgreementViewSummary>();

 Filter partyType =
Filter.builder().name(PARTY_TYPE_FILTER_NAME).values(PARTY_TYPE_FILTER_VALUE_PROPOSER).buil

 Filter agreementType =
Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME).values(AGREEMENT_TYPE_FILTER_VALUE_PURCHA

 Filter customizeFilter =
Filter.builder().name(FILTER_NAME).values(FILTER_VALUE).build();

 SearchAgreementsRequest searchAgreementsRequest =
   SearchAgreementsRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .filters(partyType, agreementType, customizeFilter).build();

 SearchAgreementsResponse searchResultResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

 agreementSummaryList.addAll(searchResultResponse.agreementViewSummaries());
```

```
 while (searchResultResponse.nextToken() != null &&
searchResultResponse.nextToken().length() > 0) {
  searchAgreementsRequest =
SearchAgreementsRequest.builder().catalog(AWS_MP_CATALOG)
    .filters(partyType,
agreementType).nextToken(searchResultResponse.nextToken()).build();
  searchResultResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
  agreementSummaryList.addAll(searchResultResponse.agreementViewSummaries());
 }
 return agreementSummaryList;

}

static class KeyPair {
 private final String first;
 private final Set<String> second;

 public KeyPair(String productType, Set<String> second) {
  this.first = productType;
  this.second = second;
 }

 @Override
 public int hashCode() {
  return Objects.hash(first, second);
 }

 @Override
 public boolean equals(Object obj) {
  if (this == obj)
   return true;
  if (obj == null || getClass() != obj.getClass())
   return false;
  KeyPair other = (KeyPair) obj;
  return Objects.equals(first, other.first) && Objects.equals(second,
other.second);
 }
}

/*
 * Get all the term types for the offer
 */
public static Set<String> getOfferTermTypes(String offerId) {
```

```
 Set<String> offerTermTypes = new HashSet<String>();

 DescribeEntityRequest request =
   DescribeEntityRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .entityId(offerId)
   .build();

 DescribeEntityResponse result = marketplaceCatalogClient.describeEntity(request);

 String details = result.details();

 try {
  ObjectMapper objectMapper = new ObjectMapper();
  JsonNode rootNode = objectMapper.readTree(details);
  JsonNode termsNode = rootNode.get(ATTRIBUTE_TERMS);

  for (JsonNode termNode : termsNode) {
   if (termNode.get(ATTRIBUTE_TYPE_ENTITY) != null ) {
    offerTermTypes.add(termNode.get(ATTRIBUTE_TYPE_ENTITY).asText());
   }
  }
 } catch (Exception e) {
  e.printStackTrace();
 }

 return offerTermTypes;

}

/*
 * Get all the agreement term types
 */
public static Set<String> getAgreementTermTypes(GetAgreementTermsResponse
agreementTerm) {
 Set<String> agreementTermTypes = new HashSet<String>();
 try {
  for (AcceptedTerm term : agreementTerm.acceptedTerms()) {
   ObjectMapper objectMapper = new ObjectMapper();
   JsonNode termNode = objectMapper.readTree(getJson(term));
   Iterator<Map.Entry<String, JsonNode>> fieldsIterator = termNode.fields();
   while (fieldsIterator.hasNext()) {
    Map.Entry<String, JsonNode> entry = fieldsIterator.next();
```

```
    JsonNode value = entry.getValue();
    if (value.isObject() && value.has(ATTRIBUTE_TYPE_AGREEMENT)) {
     agreementTermTypes.add(value.get(ATTRIBUTE_TYPE_AGREEMENT).asText());
    }
   }
  }
 } catch (Exception e) {
  e.printStackTrace();
 }
 return agreementTermTypes;

}

/*
 * make sure all elements in array2 exist in array1
 */
public static boolean allElementsExist(Set<String> array1, Set<String> array2) {
 for (String element : array2) {
  boolean found = false;
  for (String str : array1) {
   if (element.equals(str)) {
    found = true;
    break;
   }
  }
  if (!found) {
   return false;
  }
 }
 return true;
}

/*
 * Find the combinations of the agreement term types for the agreement
 */
public static Set<String> getMatchedTermTypesCombination(Set<String>
agreementTermTypes) {
 Set<String> matchedCombination = new HashSet<String>();
 for (Set<String> element : ALL_AGREEMENT_TERM_TYPES_COMBINATION) {
  if (allElementsExist(agreementTermTypes, element)) {
   matchedCombination = element;
  }
 }
 return matchedCombination;
```

```
}

public static void main(String[] args) {

 List<AgreementViewSummary> agreements = getAgreementsById();

 for (AgreementViewSummary summary : agreements) {
  String pricingType = "";
  String agreementId = summary.agreementId();
  System.out.println(agreementId);
  String offerId = summary.proposalSummary().offerId();

  //get all pricing term types for the offer in the agreement
  Set<String> offerTermTypes = getOfferTermTypes(offerId);
  String productType = summary.proposalSummary().resources().get(0).type();

  //get all pricing term types for the agreement
  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();
  GetAgreementTermsResponse getAgreementTermsResponse =
marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);
  Set<String> agreementTermTypes =
getAgreementTermTypes(getAgreementTermsResponse);

  //get matched pricing term type combination set
  Set<String> agreementMatchedTermType =
getMatchedTermTypesCombination(agreementTermTypes);

  //check to see if this agreement pricing term combination needs additional check
on offer pricing terms
  String needToCheckOfferType = needToCheckOfferTermsType(productType,
agreementMatchedTermType);

  // get the pricing type for the agreement based on the product type, agreement
term types and offer term types if needed
  if (needToCheckOfferType != null) {
   Set<String> offerMatchedTermType =
getMatchedTermTypesCombination(offerTermTypes);
   pricingType = getPricingType(productType, agreementMatchedTermType,
offerMatchedTermType);
  } else if (agreementMatchedTermType == LEGAL) {
   pricingType = FREE;
  } else {
```

```
    pricingType = getPricingType(productType, agreementMatchedTermType, new
 HashSet());
   }
   System.out.println("Pricing type is " + pricingType);
  }
 }

 private static String getJson(Object result) {
  String json = "";

  try {
   ObjectMapper om = new ObjectMapper();
   om.setVisibility(PropertyAccessor.FIELD, Visibility.ANY);
   om.registerModule(new JavaTimeModule());
   ObjectWriter ow = om.writer().withDefaultPrettyPrinter();

   json = ow.writeValueAsString(result);
  } catch (JsonProcessingException e) {
   e.printStackTrace();
  }
  return json;
 }

}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeAgreement를 참조하세요.

계약의 제품 유형 가져오기

다음 코드 예제에서는 계약의 제품 유형을 가져오는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
```

```
package com.example.awsmarketplace.agreementapi;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.Resource;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import java.util.ArrayList;
import java.util.List;

import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementProductType {

 /*
  * Obtain the Product Type of the product the agreement was created on
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<String> productIds = getProducts(agreementId);

  ReferenceCodesUtils.formatOutput(productIds);
 }

 public static List<String> getProducts(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeAgreementRequest describeAgreementRequest =
    DescribeAgreementRequest.builder()
    .agreementId(agreementId)
    .build();
```

```
  DescribeAgreementResponse describeAgreementResponse =
 marketplaceAgreementClient.describeAgreement(describeAgreementRequest);

  List<String> productIds = new ArrayList<String>();
  for (Resource resource : describeAgreementResponse.proposalSummary().resources())
 {
   productIds.add(resource.id() + ":" + resource.type());
  }
  return productIds;
 }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeAgreement를 참조하세요.

계약 상태 가져오기

다음 코드 예제에서는 계약의 상태를 가져오는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.DescribeAgreementResponse;
```

```
public class GetAgreementStatus {

 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  DescribeAgreementResponse describeAgreementResponse =
 getDescribeAgreementResponse(agreementId);

  System.out.println("Agreement status is " + describeAgreementResponse.status());

 }

 public static DescribeAgreementResponse getDescribeAgreementResponse(String
 agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  DescribeAgreementRequest describeAgreementRequest =
    DescribeAgreementRequest.builder()
    .agreementId(agreementId)
    .build();

  DescribeAgreementResponse describeAgreementResponse =
 marketplaceAgreementClient.describeAgreement(describeAgreementRequest);
  return describeAgreementResponse;
 }

}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeAgreement를 참조하세요.

계약의 지원 조건 가져오기

다음 코드 예제에서는 계약의 지원 조건을 가져오는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;
import software.amazon.awssdk.services.marketplaceagreement.model.SupportTerm;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.AGREEMENT_ID;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class GetAgreementTermsSupportTerm {

 /*
  * Obtain the support and refund policy I have provided to the customer
  */
 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  List<SupportTerm> supportTerms = getSupportTerms(agreementId);

  ReferenceCodesUtils.formatOutput(supportTerms);
 }
```

```
 public static List<SupportTerm> getSupportTerms(String agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();

  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);

  List<SupportTerm> supportTerms = new ArrayList<>();

  for (AcceptedTerm acceptedTerm : getAgreementTermsResponse.acceptedTerms()) {
   if (acceptedTerm.supportTerm() != null) {
    supportTerms.add(acceptedTerm.supportTerm());
   }
  }
  return supportTerms;
 }

}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetAgreementTerms을 참조하세요.

계약 조건 가져오기

다음 코드 예제에서는 계약 조건을 가져오는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;

public class GetAgreementTerms {

 public static void main(String[] args) {

  String agreementId = args.length > 0 ? args[0] : AGREEMENT_ID;

  GetAgreementTermsResponse getAgreementTermsResponse =
 getAgreementTermsResponse(agreementId);

  ReferenceCodesUtils.formatOutput(getAgreementTermsResponse);

 }

 public static GetAgreementTermsResponse getAgreementTermsResponse(String
 agreementId) {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder()
    .agreementId(agreementId)
    .build();
```

```
  GetAgreementTermsResponse getAgreementTermsResponse =
 marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);
  return getAgreementTermsResponse;
 }

}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetAgreementTerms을 참조하세요.

종료 날짜로 계약 검색

다음 코드 예제에서는 종료 날짜로 계약을 검색하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
```

```
import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

public class SearchAgreementsByEndDate {

 static String beforeOrAfterEndtimeFilterName =
 BeforeOrAfterEndTimeFilterName.BeforeEndTime.name();

 static String cutoffDate = "2050-11-18T00:00:00Z";

 static String partyTypeFilterValue = PARTY_TYPE_FILTER_VALUE_PROPOSER;

 public static void main(String[] args) {

  List<AgreementViewSummary> agreementSummaryList = getAgreements();

  ReferenceCodesUtils.formatOutput(agreementSummaryList);
 }

 public static List<AgreementViewSummary> getAgreements() {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  // set up filters

  Filter partyTypeFilter = Filter.builder().name(PARTY_TYPE_FILTER_NAME)
    .values(PARTY_TYPE_FILTER_VALUE_PROPOSER).build();

  Filter agreementTypeFilter = Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME)
    .values(AGREEMENT_TYPE_FILTER_VALUE_PURCHASEAGREEMENT).build();

  Filter customizeFilter =
 Filter.builder().name(beforeOrAfterEndtimeFilterName).values(cutoffDate).build();

  List<Filter> filters = new ArrayList<Filter>();

  filters.addAll(Arrays.asList(partyTypeFilter, agreementTypeFilter,
 customizeFilter));

  // search agreement with filters
```

```
 SearchAgreementsRequest searchAgreementsRequest =
   SearchAgreementsRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .filters(filters)
   .build();

 SearchAgreementsResponse searchAgreementResponse=
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

 List<AgreementViewSummary> agreementSummaryList = new
ArrayList<AgreementViewSummary>();

 agreementSummaryList.addAll(searchAgreementResponse.agreementViewSummaries());

 while (searchAgreementResponse.nextToken() != null &&
searchAgreementResponse.nextToken().length() > 0) {
  searchAgreementsRequest =
    SearchAgreementsRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .filters(filters)
    .nextToken(searchAgreementResponse.nextToken())
    .build();
  searchAgreementResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
  agreementSummaryList.addAll(searchAgreementResponse.agreementViewSummaries());
 }
 return agreementSummaryList;
}

}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 SearchAgreements를 참조하세요.

하나의 사용자 지정 필터를 적용한 계약 검색

다음 코드 예제에서는 하나의 사용자 지정 필터를 적용하여 계약을 검색하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import com.example.awsmarketplace.utils.ReferenceCodesUtils;

/**
 * To search by
 * offer id: OfferId;
 * product id: ResourceIdentifier;
 * customer AWS account id: AcceptorAccountId
 * product type: ResourceType (i.e. SaasProduct)
 * status: Status. status values can be: ACTIVE, CANCELED,
 *   EXPIRED, RENEWED, REPLACED, ROLLED_BACK, SUPERSEDED, TERMINATED
 */
```

```
public class SearchAgreementsByOneFilter {

 private static final String FILTER_NAME = "ResourceType";

 private static final String FILTER_VALUE = "SaaSProduct";

 /*
  * search agreements by one customize filter
  */
 public static void main(String[] args) {

  List<AgreementViewSummary> agreementSummaryList = getAgreements();

  ReferenceCodesUtils.formatOutput(agreementSummaryList);
 }

 public static List<AgreementViewSummary> getAgreements() {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  Filter partyTypeFilter = Filter.builder().name(PARTY_TYPE_FILTER_NAME)
    .values(PARTY_TYPE_FILTER_VALUE_PROPOSER).build();

  Filter agreementTypeFilter = Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME)
    .values(AGREEMENT_TYPE_FILTER_VALUE_PURCHASEAGREEMENT).build();

  Filter customizeFilter =
Filter.builder().name(FILTER_NAME).values(FILTER_VALUE).build();

  List<Filter> filters = new ArrayList<Filter>();

  filters.addAll(Arrays.asList(partyTypeFilter, agreementTypeFilter,
customizeFilter));

  SearchAgreementsRequest searchAgreementsRequest =
    SearchAgreementsRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .filters(filters)
    .build();
  SearchAgreementsResponse searchAgreementsResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
```

```
  List<AgreementViewSummary> agreementSummaryList = new
 ArrayList<AgreementViewSummary>();

  agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());

  while (searchAgreementsResponse.nextToken() != null &&
 searchAgreementsResponse.nextToken().length() > 0) {
   searchAgreementsRequest =
     SearchAgreementsRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .filters(filters)
     .nextToken(searchAgreementsResponse.nextToken())
     .build();
   searchAgreementsResponse =
 marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
   agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());
  }
  return agreementSummaryList;
 }

}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 SearchAgreements를 참조하세요.

두 개의 사용자 지정 필터를 적용한 계약 검색

다음 코드 예제에서는 두 개의 사용자 지정 필터를 적용하여 계약을 검색하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS Marketplace API 참조 코드 라이브러리 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;
```

```
import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

/**
 * Party Type = Proposer AND Acceptor:
 *  AfterEndTime
 *  BeforeEndTime
 *  ResourceIdentifier + BeforeEndTime
 *  ResourceIdentifier + AfterEndTime
 *  ResourceType + BeforeEndTime
 *  ResourceType + AfterEndTime
 *
 * Party Type = Proposer
 *  ResourceIdentifier
 *  OfferId
 *  AcceptorAccountId
 *  Status (ACTIVE)
 *  Status (ACTIVE) + ResourceIdentifier
 *  Status (ACTIVE) + AcceptorAccountId
 *  Status (ACTIVE) + OfferId
 *  Status (ACTIVE) + ResourceType
 *  AcceptorAccountId + BeforeEndTime
 *  AcceptorAccountId + AfterEndTime
 *  AcceptorAccountId + AfterEndTime
 *  OfferId + BeforeEndTime
 *
```

```
 * Status values can be: ACTIVE, CANCELLED, EXPIRED, RENEWED, REPLACED, ROLLED_BACK,
 SUPERSEDED, TERMINATED
 */

public class SearchAgreementsByTwoFilters {

 public static final String FILTER_1_NAME = "ResourceType";

 public static final String FILTER_1_VALUE = "SaaSProduct";

 public static final String FILTER_2_NAME = "Status";

 public static final String FILTER_2_VALUE = "ACTIVE";

 /*
  * search agreements by two customize filter
  */
 public static void main(String[] args) {

  List<AgreementViewSummary> agreementSummaryList = getAgreements();

  ReferenceCodesUtils.formatOutput(agreementSummaryList);

 }

 public static List<AgreementViewSummary> getAgreements() {
  MarketplaceAgreementClient marketplaceAgreementClient =
    MarketplaceAgreementClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  Filter partyTypeFilter = Filter.builder().name(PARTY_TYPE_FILTER_NAME)
    .values(PARTY_TYPE_FILTER_VALUE_PROPOSER).build();

  Filter agreementTypeFilter = Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME)
    .values(AGREEMENT_TYPE_FILTER_VALUE_PURCHASEAGREEMENT).build();

  Filter customizeFilter1 =
 Filter.builder().name(FILTER_1_NAME).values(FILTER_1_VALUE).build();

  Filter customizeFilter2 =
 Filter.builder().name(FILTER_2_NAME).values(FILTER_2_VALUE).build();
```

```
 List<Filter> filters = new ArrayList<Filter>();

 filters.addAll(Arrays.asList(partyTypeFilter, agreementTypeFilter,
customizeFilter1, customizeFilter2));

 SearchAgreementsRequest searchAgreementsRequest =
   SearchAgreementsRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .filters(filters)
   .build();

 SearchAgreementsResponse searchAgreementsResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

 List<AgreementViewSummary> agreementSummaryList = new
ArrayList<AgreementViewSummary>();

 agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());

 while (searchAgreementsResponse.nextToken() != null &&
searchAgreementsResponse.nextToken().length() > 0) {
  searchAgreementsRequest =
    SearchAgreementsRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .filters(filters)
    .nextToken(searchAgreementsResponse.nextToken())
    .build();
  searchAgreementsResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
  agreementSummaryList.addAll(searchAgreementsResponse.agreementViewSummaries());
 }
 return agreementSummaryList;
}

}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 SearchAgreements를 참조하세요.

Java 2.x용 SDK를 사용하는 MediaConvert 예제

다음 코드 예제에서는 MediaConvert와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`CreateJob`

다음 코드 예시는 `CreateJob`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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 {
                        System.out.println("MediaConvert role arn: " + mcRoleARN);
                        System.out.println("MediaConvert input file: " + fileInput);
                        System.out.println("MediaConvert output path: " + s3path);

                        // 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 =
mc.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;
       }
```

```
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateJob을 참조하세요.

`GetJob`

다음 코드 예시는 `GetJob`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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 {
            GetJobRequest jobRequest = GetJobRequest.builder()
                    .id(jobId)
                    .build();

            GetJobResponse response = mc.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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetJob을 참조하세요.

`ListJobs`

다음 코드 예시는 `ListJobs`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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 {
            // Create the ListJobsRequest
            ListJobsRequest jobsRequest = ListJobsRequest.builder()
```

```
                    .maxResults(10)
                    .status("COMPLETE")
                    .build();

            // Call the listJobs operation
            ListJobsResponse jobsResponse = mc.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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListJobs를 참조하세요.

Java 2.x용 SDK를 사용하는 Migration Hub 예제

다음 코드 예제에서는 Migration Hub와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`DeleteProgressUpdateStream`

다음 코드 예시는 `DeleteProgressUpdateStream`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteProgressUpdateStream을 참조하세요.

`DescribeApplicationState`

다음 코드 예시는 `DescribeApplicationState`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeApplicationState를 참조하세요.

`DescribeMigrationTask`

다음 코드 예시는 `DescribeMigrationTask`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeMigrationTask를 참조하세요.

`ImportMigrationTask`

다음 코드 예시는 `ImportMigrationTask`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ImportMigrationTask를 참조하세요.

`ListApplications`

다음 코드 예시는 `ListApplications`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListApplications를 참조하세요.

`ListCreatedArtifacts`

다음 코드 예시는 `ListCreatedArtifacts`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListCreatedArtifacts를 참조하세요.

`ListMigrationTasks`

다음 코드 예시는 `ListMigrationTasks`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListMigrationTasks를 참조하세요.

SDK for Java 2.x를 사용한 Amazon MSK 예제

다음 코드 예제에서는 Amazon MSK와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 서버리스 예제

서버리스 예제

Amazon MSK 트리거를 사용하여 Lambda 함수 간접 호출

다음 코드 예제에서는 Amazon MSK 클러스터에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 MSK 페이로드를 검색하고 레코드 콘텐츠를 로깅합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Java를 사용하여 Lambda로 Amazon MSK 이벤트를 사용합니다.

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.KafkaEvent;
import com.amazonaws.services.lambda.runtime.events.KafkaEvent.KafkaEventRecord;

import java.util.Base64;
import java.util.Map;

public class Example implements RequestHandler<KafkaEvent, Void> {

    @Override
    public Void handleRequest(KafkaEvent event, Context context) {
        for (Map.Entry<String, java.util.List<KafkaEventRecord>> entry :
 event.getRecords().entrySet()) {
            String key = entry.getKey();
            System.out.println("Key: " + key);

            for (KafkaEventRecord record : entry.getValue()) {
                System.out.println("Record: " + record);

                byte[] value = Base64.getDecoder().decode(record.getValue());
                String message = new String(value);
                System.out.println("Message: " + message);
            }
        }

        return null;
    }
}
```

SDK for Java 2.x를 사용한 Neptune 예제

다음 코드 예제에서는 Neptune과 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시작하기

Neptune 시작

다음 코드 예제에서는 Neptune 사용을 시작하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloNeptune {
    public static void main(String[] args) {
        NeptuneAsyncClient neptuneClient = NeptuneAsyncClient.create();
```

```
        describeDbCluster(neptuneClient).join(); // This ensures the async code runs
to completion
    }

    /**
     * Describes the Amazon Neptune DB clusters.
     *
     * @param neptuneClient the Neptune asynchronous client used to make the request
     * @return a {@link CompletableFuture} that completes when the operation is
finished
     */
    public static CompletableFuture<Void> describeDbCluster(NeptuneAsyncClient
neptuneClient) {
        DescribeDbClustersRequest request = DescribeDbClustersRequest.builder()
                .maxRecords(20)
                .build();

        SdkPublisher<DescribeDbClustersResponse> paginator =
neptuneClient.describeDBClustersPaginator(request);
        CompletableFuture<Void> future = new CompletableFuture<>();

        paginator.subscribe(new Subscriber<DescribeDbClustersResponse>() {
            private Subscription subscription;

            @Override
            public void onSubscribe(Subscription s) {
                this.subscription = s;
                s.request(Long.MAX_VALUE); // request all items
            }

            @Override
            public void onNext(DescribeDbClustersResponse response) {
                response.dbClusters().forEach(cluster -> {
                    System.out.println("Cluster Identifier: " +
cluster.dbClusterIdentifier());
                    System.out.println("Status: " + cluster.status());
                });
            }

            @Override
            public void onError(Throwable t) {
                future.completeExceptionally(t);
            }
```

```
            @Override
            public void onComplete() {
                future.complete(null);
            }
        });

        return future.whenComplete((result, throwable) -> {
            neptuneClient.close();
            if (throwable != null) {
                System.err.println("Error describing DB clusters: " +
throwable.getMessage());
            }
        });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeDBClustersPaginator를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- Amazon Neptune 서브넷 그룹을 만듭니다.
- Neptune 클러스터를 만듭니다.
- Neptune 인스턴스를 만듭니다.
- Neptune 인스턴스의 상태를 확인합니다.
- Neptune 클러스터 세부 정보를 표시합니다.
- Neptune 클러스터를 중지합니다.
- Neptune 클러스터를 시작합니다.
- Neptune 자산을 삭제합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Neptune 기능을 보여주는 대화형 시나리오를 실행합니다.

```
public class NeptuneScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final Logger logger =
 LoggerFactory.getLogger(NeptuneScenario.class);
    static Scanner scanner = new Scanner(System.in);
    static NeptuneActions neptuneActions = new NeptuneActions();

    public static void main(String[] args) {
        final String usage =
                """
                Usage:
                    <subnetGroupName> <clusterName> <dbInstanceId>

                Where:
                    subnetGroupName - The name of an existing Neptune DB subnet
 group that includes subnets in at least two Availability Zones.
                    clusterName     - The unique identifier for the Neptune DB
 cluster.
                    dbInstanceId    - The identifier for a specific Neptune DB
 instance within the cluster.
                """;
        String subnetGroupName = "neptuneSubnetGroup65";
        String clusterName = "neptuneCluster65";
        String dbInstanceId = "neptuneDB65";

        logger.info("""
                   Amazon Neptune is a fully managed graph
                   database service by AWS, designed specifically
                   for handling complex relationships and connected
                   datasets at scale. It supports two popular graph models:
                   property graphs (via openCypher and Gremlin) and RDF
                   graphs (via SPARQL). This makes Neptune ideal for
                   use cases such as knowledge graphs, fraud detection,
```

```
                    social networking, recommendation engines, and
                    network management, where relationships between
                    entities are central to the data.

                    Being fully managed, Neptune handles database
                    provisioning, patching, backups, and replication,
                    while also offering high availability and durability
                    within AWS's infrastructure.

                    For developers, programming with Neptune allows
                    for building intelligent, relationship-aware
                    applications that go beyond traditional tabular
                    databases. Developers can use the AWS SDK for Java
                    to automate infrastructure operations (via NeptuneClient).

                     Let's get started...
                     """);
        waitForInputToContinue(scanner);
        runScenario(subnetGroupName, dbInstanceId, clusterName);
    }

    public static void runScenario(String subnetGroupName, String dbInstanceId,
String clusterName) {
        logger.info(DASHES);
        logger.info("1. Create a Neptune DB Subnet Group");
        logger.info("The Neptune DB subnet group is used when launching a Neptune
cluster");
        waitForInputToContinue(scanner);
        try {
            neptuneActions.createSubnetGroupAsync(subnetGroupName).join();

        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ServiceQuotaExceededException) {
                logger.error("The request failed due to service quota exceeded: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);
```

```
        logger.info(DASHES);
        logger.info("2. Create a Neptune Cluster");
        logger.info("A Neptune Cluster allows you to store and query highly
connected datasets with low latency.");
        waitForInputToContinue(scanner);
        String dbClusterId;
        try {
            dbClusterId = neptuneActions.createDBClusterAsync(clusterName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ServiceQuotaExceededException) {
                logger.error("The request failed due to service quota exceeded: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Create a Neptune DB Instance");
        logger.info("In this step, we add a new database instance to the Neptune
cluster");
        waitForInputToContinue(scanner);
        try {
        neptuneActions.createDBInstanceAsync(dbInstanceId, dbClusterId).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ServiceQuotaExceededException) {
                logger.error("The request failed due to service quota exceeded: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Check the status of the Neptune DB Instance");
```

```
        logger.info("""
                    In this step, we will wait until the DB instance
                    becomes available. This may take around 10 minutes.
                    """);
        waitForInputToContinue(scanner);
        try {
            neptuneActions.checkInstanceStatus(dbInstanceId, "available").join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            logger.error("An unexpected error occurred.", cause);
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5.Show Neptune Cluster details");
        waitForInputToContinue(scanner);
        try {
            neptuneActions.describeDBClustersAsync(clusterName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The request failed due to the resource not found: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. Stop the Amazon Neptune cluster");
        logger.info("""
                    Once stopped, this step polls the status
                    until the cluster is in a stopped state.
                    """);
        waitForInputToContinue(scanner);
        try {
            neptuneActions.stopDBClusterAsync(dbClusterId);
            neptuneActions.waitForClusterStatus(dbClusterId, "stopped");
        } catch (CompletionException ce) {
```

```
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The request failed due to the resource not found: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Start the Amazon Neptune cluster");
        logger.info("""
                    Once started, this step polls the clusters
                    status until it's in an available state.
                    We will also poll the instance status.
                    """);
        waitForInputToContinue(scanner);
        try {
            neptuneActions.startDBClusterAsync(dbClusterId);
            neptuneActions.waitForClusterStatus(dbClusterId, "available");
            neptuneActions.checkInstanceStatus(dbInstanceId, "available").join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The request failed due to the resource not found: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Delete the Neptune Assets");
        logger.info("Would you like to delete the Neptune Assets? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete the Neptune assets.");
            try {
```

```
                neptuneActions.deleteNeptuneResourcesAsync(dbInstanceId,
 clusterName, subnetGroupName);
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    logger.error("The request failed due to the resource not found:
 {}", cause.getMessage());
                } else {
                    logger.error("An unexpected error occurred.", cause);
                }
                return;
            }
        } else {
            logger.info("You selected not to delete Neptune assets.");
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info(
                """
                Thank you for checking out the Amazon Neptune Service Use demo. We
 hope you
                learned something new, or got some inspiration for your own apps
 today.
                For more AWS code examples, have a look at:
                https://docs.aws.amazon.com/code-library/latest/ug/what-is-code-
library.html
                """);
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
```

```
            }
        }
    }
}
```

Neptune SDK 메서드의 래퍼 클래스입니다.

```
public class NeptuneActions {
    private CompletableFuture<Void> instanceCheckFuture;
    private static NeptuneAsyncClient neptuneAsyncClient;
    private final Region region = Region.US_EAST_1;
    private static final Logger logger =
 LoggerFactory.getLogger(NeptuneActions.class);
    private final NeptuneClient neptuneClient =
 NeptuneClient.builder().region(region).build();

    /**
     * Retrieves an instance of the NeptuneAsyncClient.
     * <p>
     * This method initializes and returns a singleton instance of the
 NeptuneAsyncClient. The client
     * is configured with the following settings:
     * <ul>
     *     <li>Maximum concurrency: 100</li>
     *     <li>Connection timeout: 60 seconds</li>
     *     <li>Read timeout: 60 seconds</li>
     *     <li>Write timeout: 60 seconds</li>
     *     <li>API call timeout: 2 minutes</li>
     *     <li>API call attempt timeout: 90 seconds</li>
     *     <li>Retry strategy: STANDARD</li>
     * </ul>
     * The client is built using the NettyNioAsyncHttpClient.
     *
     * @return the singleton instance of the NeptuneAsyncClient
     */
    private static NeptuneAsyncClient getAsyncClient() {
        if (neptuneAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                    .maxConcurrency(100)
                    .connectionTimeout(Duration.ofSeconds(60))
                    .readTimeout(Duration.ofSeconds(60))
                    .writeTimeout(Duration.ofSeconds(60))
```

```
                    .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
                    .retryStrategy(RetryMode.STANDARD)
                    .build();

            neptuneAsyncClient = NeptuneAsyncClient.builder()
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
        return neptuneAsyncClient;
    }

    /**
     * Asynchronously deletes a set of Amazon Neptune resources in a defined order.
     * <p>
     * The method performs the following operations in sequence:
     * <ol>
     *     <li>Deletes the Neptune DB instance identified by {@code dbInstanceId}.</
li>
     *     <li>Waits until the DB instance is fully deleted.</li>
     *     <li>Deletes the Neptune DB cluster identified by {@code dbClusterId}.</
li>
     *     <li>Deletes the Neptune DB subnet group identified by {@code
 subnetGroupName}.</li>
     * </ol>
     * <p>
     * If any step fails, the subsequent operations are not performed, and the
 exception
     * is logged. This method blocks the calling thread until all operations
 complete.
     *
     * @param dbInstanceId      the ID of the Neptune DB instance to delete
     * @param dbClusterId       the ID of the Neptune DB cluster to delete
     * @param subnetGroupName   the name of the Neptune DB subnet group to delete
     */
    public void deleteNeptuneResourcesAsync(String dbInstanceId, String dbClusterId,
 String subnetGroupName) {
        deleteDBInstanceAsync(dbInstanceId)
                .thenCompose(v -> waitUntilInstanceDeletedAsync(dbInstanceId))
```

```
                .thenCompose(v -> deleteDBClusterAsync(dbClusterId))
                .thenCompose(v -> deleteDBSubnetGroupAsync(subnetGroupName))
                .whenComplete((v, ex) -> {
                    if (ex != null) {
                        logger.info("Failed to delete Neptune resources: " +
ex.getMessage());
                    } else {
                        logger.info("Neptune resources deleted successfully.");
                    }
                })
                .join(); // Waits for the entire async chain to complete
    }

    /**
     * Deletes a subnet group.
     *
     * @param subnetGroupName the identifier of the subnet group to delete
     * @return a {@link CompletableFuture} that completes when the cluster has been
deleted
     */
    public CompletableFuture<Void> deleteDBSubnetGroupAsync(String subnetGroupName)
{
        DeleteDbSubnetGroupRequest request = DeleteDbSubnetGroupRequest.builder()
                .dbSubnetGroupName(subnetGroupName)
                .build();

        return getAsyncClient().deleteDBSubnetGroup(request)
                .thenAccept(response -> logger.info("## Deleting Subnet Group: " +
subnetGroupName));
    }

    /**
     * Deletes a DB instance asynchronously.
     *
     * @param clusterId the identifier of the cluster to delete
     * @return a {@link CompletableFuture} that completes when the cluster has been
deleted
     */
    public CompletableFuture<Void> deleteDBClusterAsync(String clusterId) {
        DeleteDbClusterRequest request = DeleteDbClusterRequest.builder()
                .dbClusterIdentifier(clusterId)
                .skipFinalSnapshot(true)
                .build();
```

```
        return getAsyncClient().deleteDBCluster(request)
                .thenAccept(response -> System.out.println("## Deleting DB Cluster:
" + clusterId));
    }

    public CompletableFuture<Void> waitUntilInstanceDeletedAsync(String instanceId)
{
        CompletableFuture<Void> future = new CompletableFuture<>();
        long startTime = System.currentTimeMillis();
        checkInstanceDeletedRecursive(instanceId, startTime, future);
        return future;
    }

    /**
     * Deletes a DB instance asynchronously.
     *
     * @param instanceId the identifier of the DB instance to be deleted
     * @return a {@link CompletableFuture} that completes when the DB instance has
been deleted
     */
    public CompletableFuture<Void> deleteDBInstanceAsync(String instanceId) {
        DeleteDbInstanceRequest request = DeleteDbInstanceRequest.builder()
                .dbInstanceIdentifier(instanceId)
                .skipFinalSnapshot(true)
                .build();

        return getAsyncClient().deleteDBInstance(request)
                .thenAccept(response -> System.out.println("## Deleting DB Instance:
" + instanceId));
    }


    private void checkInstanceDeletedRecursive(String instanceId, long startTime,
CompletableFuture<Void> future) {
        DescribeDbInstancesRequest request = DescribeDbInstancesRequest.builder()
                .dbInstanceIdentifier(instanceId)
                .build();

        getAsyncClient().describeDBInstances(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof NeptuneException &&
```

```
                                ((NeptuneException)
cause).awsErrorDetails().errorCode().equals("DBInstanceNotFound")) {
                            long elapsed = (System.currentTimeMillis() -
startTime) / 1000;
                            logger.info("\r Instance %s deleted after %ds%n",
instanceId, elapsed);
                            future.complete(null);
                            return;
                        }
                        future.completeExceptionally(new CompletionException("Error
polling DB instance", cause));
                        return;
                    }

                    String status =
response.dbInstances().get(0).dbInstanceStatus();
                    long elapsed = (System.currentTimeMillis() - startTime) / 1000;
                    System.out.printf("\r  Waiting: Instance %s status: %-10s (%ds
elapsed)", instanceId, status, elapsed);
                    System.out.flush();

                    CompletableFuture.delayedExecutor(20, TimeUnit.SECONDS)
                            .execute(() -> checkInstanceDeletedRecursive(instanceId,
startTime, future));
                });
    }


    public void waitForClusterStatus(String clusterId, String desiredStatus) {
        System.out.printf("Waiting for cluster '%s' to reach status '%s'...\n",
clusterId, desiredStatus);
        CompletableFuture<Void> future = new CompletableFuture<>();
        checkClusterStatusRecursive(clusterId, desiredStatus,
System.currentTimeMillis(), future);
        future.join();
    }

    private void checkClusterStatusRecursive(String clusterId, String desiredStatus,
long startTime, CompletableFuture<Void> future) {
        DescribeDbClustersRequest request = DescribeDbClustersRequest.builder()
                .dbClusterIdentifier(clusterId)
                .build();

        getAsyncClient().describeDBClusters(request)
```

```
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        future.completeExceptionally(
                                new CompletionException("Error checking Neptune
cluster status", cause)
                        );
                        return;
                    }

                    List<DBCluster> clusters = response.dbClusters();
                    if (clusters.isEmpty()) {
                        future.completeExceptionally(new RuntimeException("Cluster
not found: " + clusterId));
                        return;
                    }

                    String currentStatus = clusters.get(0).status();
                    long elapsedSeconds = (System.currentTimeMillis() - startTime) /
1000;
                    System.out.printf("\r Elapsed: %-20s  Cluster status: %-20s",
formatElapsedTime((int) elapsedSeconds), currentStatus);
                    System.out.flush();

                    if (desiredStatus.equalsIgnoreCase(currentStatus)) {
                        System.out.printf("\r Neptune cluster reached desired status
'%s' after %s.\n", desiredStatus, formatElapsedTime((int) elapsedSeconds));
                        future.complete(null);
                    } else {
                        CompletableFuture.delayedExecutor(20, TimeUnit.SECONDS)
                                .execute(() ->
checkClusterStatusRecursive(clusterId, desiredStatus, startTime, future));
                    }
                });
    }


    /**
     * Starts an Amazon Neptune DB cluster.
     *
     * @param clusterIdentifier the unique identifier of the DB cluster to be
stopped
     */
```

```
   public CompletableFuture<StartDbClusterResponse> startDBClusterAsync(String
clusterIdentifier) {
       StartDbClusterRequest clusterRequest = StartDbClusterRequest.builder()
               .dbClusterIdentifier(clusterIdentifier)
               .build();

       return getAsyncClient().startDBCluster(clusterRequest)
               .whenComplete((response, error) -> {
                   if (error != null) {
                       Throwable cause = error.getCause() != null ?
error.getCause() : error;

                       if (cause instanceof ResourceNotFoundException) {
                           throw (ResourceNotFoundException) cause;
                       }

                       throw new RuntimeException("Failed to start DB cluster: " +
cause.getMessage(), cause);
                   } else {
                       logger.info("DB Cluster starting: " + clusterIdentifier);
                   }
               });
   }

   /**
    * Stops an Amazon Neptune DB cluster.
    *
    * @param clusterIdentifier the unique identifier of the DB cluster to be
stopped
    */
   public CompletableFuture<StopDbClusterResponse> stopDBClusterAsync(String
clusterIdentifier) {
       StopDbClusterRequest clusterRequest = StopDbClusterRequest.builder()
               .dbClusterIdentifier(clusterIdentifier)
               .build();

       return getAsyncClient().stopDBCluster(clusterRequest)
               .whenComplete((response, error) -> {
                   if (error != null) {
                       Throwable cause = error.getCause() != null ?
error.getCause() : error;

                       if (cause instanceof ResourceNotFoundException) {
                           throw (ResourceNotFoundException) cause;
```

```
                        }

                        throw new RuntimeException("Failed to stop DB cluster: " +
cause.getMessage(), cause);
                    } else {
                        logger.info("DB Cluster stopped: " + clusterIdentifier);
                    }
                });
    }



    /**
     * Asynchronously describes the specified Amazon RDS DB cluster.
     *
     * @param clusterId the identifier of the DB cluster to describe
     * @return a {@link CompletableFuture} that completes when the operation is
done, or throws a {@link RuntimeException}
     * if an error occurs
     */
    public CompletableFuture<Void> describeDBClustersAsync(String clusterId) {
        DescribeDbClustersRequest request = DescribeDbClustersRequest.builder()
                .dbClusterIdentifier(clusterId)
                .build();

        return getAsyncClient().describeDBClusters(request)
                .thenAccept(response -> {
                    for (DBCluster cluster : response.dbClusters()) {
                        logger.info("Cluster Identifier: " +
cluster.dbClusterIdentifier());
                        logger.info("Status: " + cluster.status());
                        logger.info("Engine: " + cluster.engine());
                        logger.info("Engine Version: " + cluster.engineVersion());
                        logger.info("Endpoint: " + cluster.endpoint());
                        logger.info("Reader Endpoint: " + cluster.readerEndpoint());
                        logger.info("Availability Zones: " +
cluster.availabilityZones());
                        logger.info("Subnet Group: " + cluster.dbSubnetGroup());
                        logger.info("VPC Security Groups:");
                        cluster.vpcSecurityGroups().forEach(vpcGroup ->
                                logger.info("  - " +
vpcGroup.vpcSecurityGroupId()));
                        logger.info("Storage Encrypted: " +
cluster.storageEncrypted());
```

```
                        logger.info("IAM DB Auth Enabled: " +
cluster.iamDatabaseAuthenticationEnabled());
                        logger.info("Backup Retention Period: " +
cluster.backupRetentionPeriod() + " days");
                        logger.info("Preferred Backup Window: " +
cluster.preferredBackupWindow());
                        logger.info("Preferred Maintenance Window: " +
cluster.preferredMaintenanceWindow());
                        logger.info("------");
                    }
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ResourceNotFoundException) {
                        throw (ResourceNotFoundException) cause;
                    }

                    throw new RuntimeException("Failed to describe the DB cluster: "
+ cause.getMessage(), cause);
                });
    }


    public CompletableFuture<Void> checkInstanceStatus(String instanceId, String
desiredStatus) {
        CompletableFuture<Void> future = new CompletableFuture<>();
        long startTime = System.currentTimeMillis();
        checkStatusRecursive(instanceId, desiredStatus.toLowerCase(), startTime,
future);
        return future;
    }

    /**
     * Checks the status of a Neptune instance recursively until the desired status
is reached or a timeout occurs.
     *
     * @param instanceId     the ID of the Neptune instance to check
     * @param desiredStatus  the desired status of the Neptune instance
     * @param startTime      the start time of the operation, used to calculate the
elapsed time
     * @param future         a {@link CompletableFuture} that will be completed when
the desired status is reached
     */
```

```
   private void checkStatusRecursive(String instanceId, String desiredStatus, long
startTime, CompletableFuture<Void> future) {
       DescribeDbInstancesRequest request = DescribeDbInstancesRequest.builder()
               .dbInstanceIdentifier(instanceId)
               .build();

       getAsyncClient().describeDBInstances(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       future.completeExceptionally(
                               new CompletionException("Error checking Neptune
instance status", cause)
                       );
                       return;
                   }

                   List<DBInstance> instances = response.dbInstances();
                   if (instances.isEmpty()) {
                       future.completeExceptionally(new RuntimeException("Instance
not found: " + instanceId));
                       return;
                   }

                   String currentStatus = instances.get(0).dbInstanceStatus();
                   long elapsedSeconds = (System.currentTimeMillis() - startTime) /
1000;
                   System.out.printf("\r Elapsed: %-20s  Status: %-20s",
formatElapsedTime((int) elapsedSeconds), currentStatus);
                   System.out.flush();

                   if (desiredStatus.equalsIgnoreCase(currentStatus)) {
                       System.out.printf("\r Neptune instance reached desired
status '%s' after %s.\n", desiredStatus, formatElapsedTime((int) elapsedSeconds));
                       future.complete(null);
                   } else {
                       CompletableFuture.delayedExecutor(20, TimeUnit.SECONDS)
                               .execute(() -> checkStatusRecursive(instanceId,
desiredStatus, startTime, future));
                   }
               });
   }
```

```
    private String formatElapsedTime(int seconds) {
        int minutes = seconds / 60;
        int remainingSeconds = seconds % 60;

        if (minutes > 0) {
            return minutes + (minutes == 1 ? " min" : " mins") + ", " +
                    remainingSeconds + (remainingSeconds == 1 ? " sec" : " secs");
        } else {
            return remainingSeconds + (remainingSeconds == 1 ? " sec" : " secs");
        }
    }


    /**
     * Creates a new Amazon Neptune DB instance asynchronously.
     *
     * @param dbInstanceId the identifier for the new DB instance
     * @param dbClusterId  the identifier for the DB cluster that the new instance
will be a part of
     * @return a {@link CompletableFuture} that completes with the identifier of the
newly created DB instance
     * @throws CompletionException if the operation fails, with a cause of either:
     *                             - {@link ServiceQuotaExceededException} if the
request would exceed the maximum quota, or
     *                             - a general exception with the failure message
     */
    public CompletableFuture<String> createDBInstanceAsync(String dbInstanceId,
String dbClusterId) {
        CreateDbInstanceRequest request = CreateDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceId)
                .dbInstanceClass("db.r5.large")
                .engine("neptune")
                .dbClusterIdentifier(dbClusterId)
                .build();

        return getAsyncClient().createDBInstance(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ServiceQuotaExceededException) {
                            throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                        }
```

```
                        throw new CompletionException("Failed to create Neptune DB
instance: " + exception.getMessage(), exception);
                    }
                })
                .thenApply(response -> {
                    String instanceId =
response.dbInstance().dbInstanceIdentifier();
                    logger.info("Created Neptune DB Instance: " + instanceId);
                    return instanceId;
                });
    }


    /**
     * Creates a new Amazon Neptune DB cluster asynchronously.
     *
     * @param dbName the name of the DB cluster to be created
     * @return a CompletableFuture that, when completed, provides the ID of the
created DB cluster
     * @throws CompletionException if the operation fails for any reason, including
if the request would exceed the maximum quota
     */
    public CompletableFuture<String> createDBClusterAsync(String dbName) {
        CreateDbClusterRequest request = CreateDbClusterRequest.builder()
                .dbClusterIdentifier(dbName)
                .engine("neptune")
                .deletionProtection(false)
                .backupRetentionPeriod(1)
                .build();

        return getAsyncClient().createDBCluster(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ServiceQuotaExceededException) {
                            throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                        }
                        throw new CompletionException("Failed to create Neptune DB
cluster: " + exception.getMessage(), exception);
                    }
                })
                .thenApply(response -> {
                    String clusterId = response.dbCluster().dbClusterIdentifier();
```

```
                    logger.info("DB Cluster created: " + clusterId);
                    return clusterId;
                });
    }


    /**
     * Creates a new DB subnet group asynchronously.
     *
     * @param groupName the name of the subnet group to create
     * @return a CompletableFuture that, when completed, returns the Amazon Resource
Name (ARN) of the created subnet group
     * @throws CompletionException if the operation fails, with a cause that may be
a ServiceQuotaExceededException if the request would exceed the maximum quota
     */
    public CompletableFuture<String> createSubnetGroupAsync(String groupName) {

        // Get the Amazon Virtual Private Cloud (VPC) where the Neptune cluster and
resources will be created
        String vpcId = getDefaultVpcId();
        logger.info("VPC is : " + vpcId);

        List<String> subnetList = getSubnetIds(vpcId);
        for (String subnetId : subnetList) {
            System.out.println("Subnet group:" +subnetId);
        }

        CreateDbSubnetGroupRequest request = CreateDbSubnetGroupRequest.builder()
                .dbSubnetGroupName(groupName)
                .dbSubnetGroupDescription("Subnet group for Neptune cluster")
                .subnetIds(subnetList)
                .build();

        return getAsyncClient().createDBSubnetGroup(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ServiceQuotaExceededException) {
                            throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                        }
                        throw new CompletionException("Failed to create subnet
group: " + exception.getMessage(), exception);
                    }
```

```
                })
                .thenApply(response -> {
                    String name = response.dbSubnetGroup().dbSubnetGroupName();
                    String arn = response.dbSubnetGroup().dbSubnetGroupArn();
                    logger.info("Subnet group created: " + name);
                    return arn;
                });
    }

    private List<String> getSubnetIds(String vpcId) {
        try (Ec2Client ec2 = Ec2Client.builder().region(region).build()) {
            DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
                    .filters(builder -> builder.name("vpc-id").values(vpcId))
                    .build();

            DescribeSubnetsResponse response = ec2.describeSubnets(request);
            return response.subnets().stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());
        }
    }

    public static String getDefaultVpcId() {
        Ec2Client ec2 = Ec2Client.builder()
                .region(Region.US_EAST_1)
                .build();

        Filter myFilter = Filter.builder()
                .name("isDefault")
                .values("true")
                .build();

        List<Filter> filterList = new ArrayList<>();
        filterList.add(myFilter);

        DescribeVpcsRequest request = DescribeVpcsRequest.builder()
                .filters(filterList)
                .build();


        DescribeVpcsResponse response = ec2.describeVpcs(request);
        if (!response.vpcs().isEmpty()) {
            Vpc defaultVpc = response.vpcs().get(0);
            return defaultVpc.vpcId();
```

```
        } else {
            throw new RuntimeException("No default VPC found in this region.");
        }
    }
}
```

작업

`CreateDBCluster`

다음 코드 예시는 `CreateDBCluster`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Creates a new Amazon Neptune DB cluster asynchronously.
    *
    * @param dbName the name of the DB cluster to be created
    * @return a CompletableFuture that, when completed, provides the ID of the
created DB cluster
    * @throws CompletionException if the operation fails for any reason, including
if the request would exceed the maximum quota
    */
   public CompletableFuture<String> createDBClusterAsync(String dbName) {
       CreateDbClusterRequest request = CreateDbClusterRequest.builder()
               .dbClusterIdentifier(dbName)
               .engine("neptune")
               .deletionProtection(false)
               .backupRetentionPeriod(1)
               .build();

       return getAsyncClient().createDBCluster(request)
               .whenComplete((response, exception) -> {
```

```
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ServiceQuotaExceededException) {
                            throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                        }
                        throw new CompletionException("Failed to create Neptune DB
cluster: " + exception.getMessage(), exception);
                    }
                })
                .thenApply(response -> {
                    String clusterId = response.dbCluster().dbClusterIdentifier();
                    logger.info("DB Cluster created: " + clusterId);
                    return clusterId;
                });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateDBCluster를 참조하세요.

`CreateDBInstance`

다음 코드 예시는 `CreateDBInstance`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Creates a new Amazon Neptune DB instance asynchronously.
     *
     * @param dbInstanceId the identifier for the new DB instance
     * @param dbClusterId  the identifier for the DB cluster that the new instance
will be a part of
     * @return a {@link CompletableFuture} that completes with the identifier of the
newly created DB instance
```

```
    * @throws CompletionException if the operation fails, with a cause of either:
    *                             - {@link ServiceQuotaExceededException} if the
request would exceed the maximum quota, or
    *                             - a general exception with the failure message
    */
   public CompletableFuture<String> createDBInstanceAsync(String dbInstanceId,
String dbClusterId) {
       CreateDbInstanceRequest request = CreateDbInstanceRequest.builder()
               .dbInstanceIdentifier(dbInstanceId)
               .dbInstanceClass("db.r5.large")
               .engine("neptune")
               .dbClusterIdentifier(dbClusterId)
               .build();

       return getAsyncClient().createDBInstance(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof ServiceQuotaExceededException) {
                           throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                       }
                       throw new CompletionException("Failed to create Neptune DB
instance: " + exception.getMessage(), exception);
                   }
               })
               .thenApply(response -> {
                   String instanceId =
response.dbInstance().dbInstanceIdentifier();
                   logger.info("Created Neptune DB Instance: " + instanceId);
                   return instanceId;
               });
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateDBInstance를 참조하세요.

`CreateDBSubnetGroup`

다음 코드 예시는 `CreateDBSubnetGroup`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Creates a new DB subnet group asynchronously.
    *
    * @param groupName the name of the subnet group to create
    * @return a CompletableFuture that, when completed, returns the Amazon Resource
Name (ARN) of the created subnet group
    * @throws CompletionException if the operation fails, with a cause that may be
a ServiceQuotaExceededException if the request would exceed the maximum quota
    */
   public CompletableFuture<String> createSubnetGroupAsync(String groupName) {

       // Get the Amazon Virtual Private Cloud (VPC) where the Neptune cluster and
resources will be created
       String vpcId = getDefaultVpcId();
       logger.info("VPC is : " + vpcId);

       List<String> subnetList = getSubnetIds(vpcId);
       for (String subnetId : subnetList) {
           System.out.println("Subnet group:" +subnetId);
       }

       CreateDbSubnetGroupRequest request = CreateDbSubnetGroupRequest.builder()
               .dbSubnetGroupName(groupName)
               .dbSubnetGroupDescription("Subnet group for Neptune cluster")
               .subnetIds(subnetList)
               .build();

       return getAsyncClient().createDBSubnetGroup(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof ServiceQuotaExceededException) {
```

```
                            throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                        }
                        throw new CompletionException("Failed to create subnet
group: " + exception.getMessage(), exception);
                    }
                })
                .thenApply(response -> {
                    String name = response.dbSubnetGroup().dbSubnetGroupName();
                    String arn = response.dbSubnetGroup().dbSubnetGroupArn();
                    logger.info("Subnet group created: " + name);
                    return arn;
                });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateDBSubnetGroup을 참조하세요.

`CreateGraph`

다음 코드 예시는 `CreateGraph`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Executes the process of creating a new Neptune graph.
     *
     * @param client        the Neptune graph client used to interact with the
Neptune service
     * @param graphName     the name of the graph to be created
     * @throws NeptuneGraphException if an error occurs while creating the graph
     */
    public static void executeCreateGraph(NeptuneGraphClient client, String
graphName) {
        try {
            // Create the graph request
```

```
            CreateGraphRequest request = CreateGraphRequest.builder()
                    .graphName(graphName)
                    .provisionedMemory(16)
                    .build();

            // Create the graph
            CreateGraphResponse response = client.createGraph(request);

            // Extract the graph name and ARN
            String createdGraphName = response.name();
            String graphArn = response.arn();
            String graphEndpoint = response.endpoint();

            System.out.println("Graph created successfully!");
            System.out.println("Graph Name: " + createdGraphName);
            System.out.println("Graph ARN: " + graphArn);
            System.out.println("Graph Endpoint: " +graphEndpoint );

        } catch (NeptuneGraphException e) {
            System.err.println("Failed to create graph: " +
e.awsErrorDetails().errorMessage());
        } finally {
            client.close();
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateGraph를 참조하세요.

`DeleteDBCluster`

다음 코드 예시는 `DeleteDBCluster`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
```

```
    * Deletes a DB instance asynchronously.
    *
    * @param clusterId the identifier of the cluster to delete
    * @return a {@link CompletableFuture} that completes when the cluster has been
deleted
    */
   public CompletableFuture<Void> deleteDBClusterAsync(String clusterId) {
       DeleteDbClusterRequest request = DeleteDbClusterRequest.builder()
               .dbClusterIdentifier(clusterId)
               .skipFinalSnapshot(true)
               .build();

       return getAsyncClient().deleteDBCluster(request)
               .thenAccept(response -> System.out.println("## Deleting DB Cluster:
" + clusterId));
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteDBCluster를 참조하세요.

`DeleteDBInstance`

다음 코드 예시는 `DeleteDBInstance`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Deletes a DB instance asynchronously.
    *
    * @param instanceId the identifier of the DB instance to be deleted
    * @return a {@link CompletableFuture} that completes when the DB instance has
been deleted
    */
   public CompletableFuture<Void> deleteDBInstanceAsync(String instanceId) {
       DeleteDbInstanceRequest request = DeleteDbInstanceRequest.builder()
               .dbInstanceIdentifier(instanceId)
```

```
                .skipFinalSnapshot(true)
                .build();

        return getAsyncClient().deleteDBInstance(request)
                .thenAccept(response -> System.out.println("## Deleting DB Instance:
" + instanceId));
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteDBInstance를 참조하세요.

`DeleteDBSubnetGroup`

다음 코드 예시는 `DeleteDBSubnetGroup`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Deletes a subnet group.
     *
     * @param subnetGroupName the identifier of the subnet group to delete
     * @return a {@link CompletableFuture} that completes when the cluster has been
deleted
     */
    public CompletableFuture<Void> deleteDBSubnetGroupAsync(String subnetGroupName)
{
        DeleteDbSubnetGroupRequest request = DeleteDbSubnetGroupRequest.builder()
                .dbSubnetGroupName(subnetGroupName)
                .build();

        return getAsyncClient().deleteDBSubnetGroup(request)
                .thenAccept(response -> logger.info("## Deleting Subnet Group: " +
subnetGroupName));
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteDBSubnetGroup을 참조하세요.

`DescribeDBClusters`

다음 코드 예시는 `DescribeDBClusters`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Asynchronously describes the specified Amazon RDS DB cluster.
    *
    * @param clusterId the identifier of the DB cluster to describe
    * @return a {@link CompletableFuture} that completes when the operation is
done, or throws a {@link RuntimeException}
    * if an error occurs
    */
   public CompletableFuture<Void> describeDBClustersAsync(String clusterId) {
       DescribeDbClustersRequest request = DescribeDbClustersRequest.builder()
               .dbClusterIdentifier(clusterId)
               .build();

       return getAsyncClient().describeDBClusters(request)
               .thenAccept(response -> {
                   for (DBCluster cluster : response.dbClusters()) {
                       logger.info("Cluster Identifier: " +
cluster.dbClusterIdentifier());
                       logger.info("Status: " + cluster.status());
                       logger.info("Engine: " + cluster.engine());
                       logger.info("Engine Version: " + cluster.engineVersion());
                       logger.info("Endpoint: " + cluster.endpoint());
                       logger.info("Reader Endpoint: " + cluster.readerEndpoint());
                       logger.info("Availability Zones: " +
cluster.availabilityZones());
                       logger.info("Subnet Group: " + cluster.dbSubnetGroup());
                       logger.info("VPC Security Groups:");
```

```
                        cluster.vpcSecurityGroups().forEach(vpcGroup ->
                                logger.info("  - " +
vpcGroup.vpcSecurityGroupId()));
                        logger.info("Storage Encrypted: " +
cluster.storageEncrypted());
                        logger.info("IAM DB Auth Enabled: " +
cluster.iamDatabaseAuthenticationEnabled());
                        logger.info("Backup Retention Period: " +
cluster.backupRetentionPeriod() + " days");
                        logger.info("Preferred Backup Window: " +
cluster.preferredBackupWindow());
                        logger.info("Preferred Maintenance Window: " +
cluster.preferredMaintenanceWindow());
                        logger.info("------");
                    }
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ResourceNotFoundException) {
                        throw (ResourceNotFoundException) cause;
                    }

                    throw new RuntimeException("Failed to describe the DB cluster: "
+ cause.getMessage(), cause);
                });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeDBClusters를 참조하세요.

`DescribeDBInstances`

다음 코드 예시는 `DescribeDBInstances`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Checks the status of a Neptune instance recursively until the desired status
is reached or a timeout occurs.
    *
    * @param instanceId     the ID of the Neptune instance to check
    * @param desiredStatus  the desired status of the Neptune instance
    * @param startTime      the start time of the operation, used to calculate the
elapsed time
    * @param future         a {@link CompletableFuture} that will be completed when
the desired status is reached
    */
   private void checkStatusRecursive(String instanceId, String desiredStatus, long
startTime, CompletableFuture<Void> future) {
       DescribeDbInstancesRequest request = DescribeDbInstancesRequest.builder()
               .dbInstanceIdentifier(instanceId)
               .build();

       getAsyncClient().describeDBInstances(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       future.completeExceptionally(
                               new CompletionException("Error checking Neptune
instance status", cause)
                       );
                       return;
                   }

                   List<DBInstance> instances = response.dbInstances();
                   if (instances.isEmpty()) {
                       future.completeExceptionally(new RuntimeException("Instance
not found: " + instanceId));
                       return;
                   }

                   String currentStatus = instances.get(0).dbInstanceStatus();
                   long elapsedSeconds = (System.currentTimeMillis() - startTime) /
1000;
                   System.out.printf("\r Elapsed: %-20s  Status: %-20s",
formatElapsedTime((int) elapsedSeconds), currentStatus);
                   System.out.flush();

                   if (desiredStatus.equalsIgnoreCase(currentStatus)) {
```

```
                        System.out.printf("\r Neptune instance reached desired
status '%s' after %s.\n", desiredStatus, formatElapsedTime((int) elapsedSeconds));
                        future.complete(null);
                    } else {
                        CompletableFuture.delayedExecutor(20, TimeUnit.SECONDS)
                                .execute(() -> checkStatusRecursive(instanceId,
desiredStatus, startTime, future));
                    }
                });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeDBInstances를 참조하세요.

`ExecuteGremlinProfileQuery`

다음 코드 예시는 `ExecuteGremlinProfileQuery`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Executes a Gremlin query against an Amazon Neptune database using the
provided {@link NeptunedataClient}.
     *
     * @param client the {@link NeptunedataClient} instance to use for executing the
Gremlin query
     */
    public static void executeGremlinQuery(NeptunedataClient client) {
        try {
            System.out.println("Querying Neptune...");
            ExecuteGremlinQueryRequest request =
ExecuteGremlinQueryRequest.builder()
                    .gremlinQuery("g.V().has('code', 'ANC')")
                    .build();
```

```
            ExecuteGremlinQueryResponse response =
client.executeGremlinQuery(request);

            System.out.println("Full Response:");
            System.out.println(response);

            // Retrieve and print the result
            if (response.result() != null) {
                System.out.println("Query Result:");
                System.out.println(response.result().toString());
            } else {
                System.out.println("No result returned from the query.");
            }
        } catch (NeptunedataException e) {
            System.err.println("Error calling Neptune: " +
e.awsErrorDetails().errorMessage());
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
        } finally {
            client.close();
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ExecuteGremlinProfileQuery를 참조하세요.

`ExecuteGremlinQuery`

다음 코드 예시는 `ExecuteGremlinQuery`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Executes a Gremlin PROFILE query using the provided NeptunedataClient.
     *
```

```
     * @param client The NeptunedataClient instance to be used for executing the
Gremlin PROFILE query.
     */
    private static void executeGremlinProfileQuery(NeptunedataClient client) {
        System.out.println("Executing Gremlin PROFILE query...");

        ExecuteGremlinProfileQueryRequest request =
ExecuteGremlinProfileQueryRequest.builder()
                .gremlinQuery("g.V().has('code', 'ANC')")
                .build();

        ExecuteGremlinProfileQueryResponse response =
client.executeGremlinProfileQuery(request);
        if (response.output() != null) {
            System.out.println("Query Profile Output:");
            System.out.println(response.output());
        } else {
            System.out.println("No output returned from the profile query.");
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ExecuteGremlinQuery를 참조하세요.

`ExecuteOpenCypherExplainQuery`

다음 코드 예시는 `ExecuteOpenCypherExplainQuery`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Executes an OpenCypher EXPLAIN query using the provided Neptune data client.
     *
     * @param client The Neptune data client to use for the query execution.
     */
```

```
    public static void executeGremlinQuery(NeptunedataClient client) {
        try {
            System.out.println("Executing OpenCypher EXPLAIN query...");
            ExecuteOpenCypherExplainQueryRequest request =
ExecuteOpenCypherExplainQueryRequest.builder()
                    .openCypherQuery("MATCH (n {code: 'ANC'}) RETURN n")
                    .explainMode("debug")
                    .build();

            ExecuteOpenCypherExplainQueryResponse response =
client.executeOpenCypherExplainQuery(request);

            if (response.results() != null) {
                System.out.println("Explain Results:");
                System.out.println(response.results().asUtf8String());
            } else {
                System.out.println("No explain results returned.");
            }

        } catch (NeptunedataException e) {
            System.err.println("Neptune error: " +
e.awsErrorDetails().errorMessage());
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
        } finally {
            client.close();
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ExecuteOpenCypherExplainQuery를 참조하세요.

`ExecuteQuery`

다음 코드 예시는 `ExecuteQuery`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Executes a Gremlin profile query on the Neptune Analytics graph.
    *
    * @param client       the {@link NeptuneGraphClient} instance to use for the
query
    * @param graphId      the identifier of the graph to execute the query on
    *
    * @throws NeptuneGraphException if an error occurs while executing the query on
the Neptune Graph
    * @throws Exception if an unexpected error occurs
    */
   public static void executeGremlinProfileQuery(NeptuneGraphClient client, String
graphId) {

       try {
           System.out.println("Running openCypher query on Neptune Analytics...");

           ExecuteQueryRequest request = ExecuteQueryRequest.builder()
                   .graphIdentifier(graphId)
                   .queryString("MATCH (n {code: 'ANC'}) RETURN n")
                   .language("OPEN_CYPHER")
                   .build();

           ResponseInputStream<ExecuteQueryResponse> response =
client.executeQuery(request);
           try (BufferedReader reader = new BufferedReader(new
InputStreamReader(response, StandardCharsets.UTF_8))) {
               String result = reader.lines().collect(Collectors.joining("\n"));
               System.out.println("Query Result:");
               System.out.println(result);
           } catch (Exception e) {
               System.err.println("Error reading response: " + e.getMessage());
           }
```

```
        } catch (NeptuneGraphException e) {
            System.err.println("NeptuneGraph error: " +
e.awsErrorDetails().errorMessage());
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
        } finally {
            client.close();
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ExecuteQuery를 참조하세요.

`StartDBCluster`

다음 코드 예시는 `StartDBCluster`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Starts an Amazon Neptune DB cluster.
     *
     * @param clusterIdentifier the unique identifier of the DB cluster to be
stopped
     */
    public CompletableFuture<StartDbClusterResponse> startDBClusterAsync(String
clusterIdentifier) {
        StartDbClusterRequest clusterRequest = StartDbClusterRequest.builder()
                .dbClusterIdentifier(clusterIdentifier)
                .build();

        return getAsyncClient().startDBCluster(clusterRequest)
                .whenComplete((response, error) -> {
                    if (error != null) {
                        Throwable cause = error.getCause() != null ?
error.getCause() : error;
```

```
                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }

                        throw new RuntimeException("Failed to start DB cluster: " +
cause.getMessage(), cause);
                    } else {
                        logger.info("DB Cluster starting: " + clusterIdentifier);
                    }
                });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartDBCluster를 참조하세요.

`StopDBCluster`

다음 코드 예시는 `StopDBCluster`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Stops an Amazon Neptune DB cluster.
     *
     * @param clusterIdentifier the unique identifier of the DB cluster to be
stopped
     */
    public CompletableFuture<StopDbClusterResponse> stopDBClusterAsync(String
clusterIdentifier) {
        StopDbClusterRequest clusterRequest = StopDbClusterRequest.builder()
                .dbClusterIdentifier(clusterIdentifier)
                .build();

        return getAsyncClient().stopDBCluster(clusterRequest)
                .whenComplete((response, error) -> {
```

```
                    if (error != null) {
                        Throwable cause = error.getCause() != null ?
error.getCause() : error;

                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }

                        throw new RuntimeException("Failed to stop DB cluster: " +
cause.getMessage(), cause);
                    } else {
                        logger.info("DB Cluster stopped: " + clusterIdentifier);
                    }
                });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StopDBCluster를 참조하세요.

시나리오

Neptune API를 사용하여 그래프 데이터 쿼리

다음 코드 예제에서는 Neptune API를 사용하여 그래프 데이터를 쿼리하는 방법을 보여줍니다.

SDK for Java 2.x

Amazon Neptune Java API를 사용하여 VPC 내에서 그래프 데이터를 쿼리하는 Lambda 함수를 생성하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Lambda
- Neptune

SDK for Java 2.x를 사용한 Partner Central 예제

다음 코드 예제에서는 Partner Central과 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`AssignOpportunity`

다음 코드 예시는 `AssignOpportunity`의 사용 방법을 보여줍니다.

SDK for Java 2.x

기존 기회를 다른 사용자에게 재할당합니다.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssignOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssignOpportunityResponse;
import software.amazon.awssdk.services.partnercentralselling.model.AssigneeContact;

/*
```

```
Purpose
PC-API-07 Assigning a new owner
*/

public class AssignOpportunity {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     String assigneeFirstName = "John";

     String assigneeLastName = "Doe";

     String assigneeEmail = "test@test.com";

     String businessTitle = "PartnerAccountManager";

     AssignOpportunityResponse response = getResponse(opportunityId,
 assigneeFirstName, assigneeLastName, assigneeEmail, businessTitle);

     ReferenceCodesUtils.formatOutput(response);
    }

 static AssignOpportunityResponse getResponse(String opportunityId, String
 assigneeFirstName, String assigneeLastName, String assigneeEmail, String
 businessTitle) {

  AssignOpportunityRequest assignOpportunityRequest =
 AssignOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
          .assignee(AssigneeContact.builder()
            .firstName(assigneeFirstName)
            .lastName(assigneeLastName)
            .email(assigneeEmail)
            .businessTitle(businessTitle)
            .build())
```

```
            .build();

        AssignOpportunityResponse response =
 client.assignOpportunity(assignOpportunityRequest);

        return response;
 }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 AssignOpportunity를 참조하세요.

`AssociateOpportunity`

다음 코드 예시는 `AssociateOpportunity`의 사용 방법을 보여줍니다.

SDK for Java 2.x

기회와 다양한 관련 엔터티 간에 공식 연결을 생성합니다.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityResponse;

/*
Purpose
PC-API -11 Associating a product
PC-API -12 Associating a solution
PC-API -13 Associating an offer
entity_type = Solutions | AWSProducts | AWSMarketplaceOffers
```

```
*/

public class AssociateOpportunity {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     String entityType = "Solutions";

     String entityIdentifier = "S-0000000";

     AssociateOpportunityResponse response = getResponse(opportunityId, entityType,
 entityIdentifier );

     ReferenceCodesUtils.formatOutput(response);
    }

static AssociateOpportunityResponse getResponse(String opportunityId, String
 entityType, String entityIdentifier) {

        AssociateOpportunityRequest associateOpportunityRequest =
 AssociateOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .opportunityIdentifier(opportunityId)
          .relatedEntityType(entityType)
          .relatedEntityIdentifier(entityIdentifier)
          .build();

        AssociateOpportunityResponse response =
 client.associateOpportunity(associateOpportunityRequest);

        return response;
 }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 AssociateOpportunity를 참조하세요.

`CreateOpportunity`

다음 코드 예시는 `CreateOpportunity`의 사용 방법을 보여줍니다.

SDK for Java 2.x

기회를 생성합니다.

```
package org.example;

import java.time.Instant;
import java.util.ArrayList;
import java.util.List;

import static org.example.utils.Constants.*;

import org.example.entity.Root;
import org.example.utils.ReferenceCodesUtils;
import org.example.utils.StringSerializer;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import software.amazon.awssdk.services.partnercentralselling.model.Account;
import software.amazon.awssdk.services.partnercentralselling.model.Address;
import software.amazon.awssdk.services.partnercentralselling.model.Contact;
import
 software.amazon.awssdk.services.partnercentralselling.model.CreateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.CreateOpportunityResponse;
import software.amazon.awssdk.services.partnercentralselling.model.Customer;
import
 software.amazon.awssdk.services.partnercentralselling.model.ExpectedCustomerSpend;
import software.amazon.awssdk.services.partnercentralselling.model.LifeCycle;
import software.amazon.awssdk.services.partnercentralselling.model.Marketing;
import software.amazon.awssdk.services.partnercentralselling.model.MonetaryValue;
import software.amazon.awssdk.services.partnercentralselling.model.NextStepsHistory;
import software.amazon.awssdk.services.partnercentralselling.model.Project;
import software.amazon.awssdk.services.partnercentralselling.model.SoftwareRevenue;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
```

```
import com.google.gson.ToNumberPolicy;

public class CreateOpportunity {

 static final Gson GSON = new GsonBuilder()
   .setObjectToNumberStrategy(ToNumberPolicy.LAZILY_PARSED_NUMBER)
   .registerTypeAdapter(String.class, new StringSerializer())
   .create();

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

 public static void main(String[] args) {

  String inputFile = "CreateOpportunity2.json";

  if (args.length > 0)
   inputFile = args[0];

  CreateOpportunityResponse response = createOpportunity(inputFile);

  client.close();
 }

 static CreateOpportunityResponse createOpportunity(String inputFile) {

  String inputString = ReferenceCodesUtils.readInputFileToString(inputFile);

  Root root = GSON.fromJson(inputString, Root.class);

  List<NextStepsHistory> nextStepsHistories = new ArrayList<NextStepsHistory>();
  if ( root.lifeCycle != null && root.lifeCycle.nextStepsHistories != null) {
   for (org.example.entity.NextStepsHistory nextStepsHistoryJson :
 root.lifeCycle.nextStepsHistories) {
    NextStepsHistory nextStepsHistory = NextStepsHistory.builder()
      .time(Instant.parse(nextStepsHistoryJson.time))
      .value(nextStepsHistoryJson.value)
                 .build();
    nextStepsHistories.add(nextStepsHistory);
   }
  }
```

```
 LifeCycle lifeCycle = null;
 if ( root.lifeCycle != null ) {
  lifeCycle = LifeCycle.builder()
    .closedLostReason(root.lifeCycle.closedLostReason)
    .nextSteps(root.lifeCycle.nextSteps)
    .nextStepsHistory(nextStepsHistories)
    .reviewComments(root.lifeCycle.reviewComments)
    .reviewStatus(root.lifeCycle.reviewStatus)
    .reviewStatusReason(root.lifeCycle.reviewStatusReason)
    .stage(root.lifeCycle.stage)
    .targetCloseDate(root.lifeCycle.targetCloseDate)
    .build();
 }

 Marketing marketing = null;
 if ( root.marketing != null ) {
  marketing = Marketing.builder()
     .awsFundingUsed(root.marketing.awsFundingUsed)
     .campaignName(root.marketing.campaignName)
     .channels(root.marketing.channels)
     .source(root.marketing.source)
     .useCases(root.marketing.useCases)
     .build();

 }

 Address address = null;
 if ( root.customer != null && root.customer.account != null &&
root.customer.account.address != null ) {
  address = Address.builder()
    .city(root.customer.account.address.city)
                .postalCode(root.customer.account.address.postalCode)
                .stateOrRegion(root.customer.account.address.stateOrRegion)
                .countryCode(root.customer.account.address.countryCode)
                .streetAddress(root.customer.account.address.streetAddress)
                .build();
 }

 Account account = null;
 if ( root.customer != null && root.customer.account!= null) {
  account = Account.builder()
             .address(address)
             .awsAccountId(root.customer.account.awsAccountId)
```

```
                .duns(root.customer.account.duns)
                .industry(root.customer.account.industry)
                .otherIndustry(root.customer.account.otherIndustry)
                .companyName(root.customer.account.companyName)
                .websiteUrl(root.customer.account.websiteUrl)
                .build();
 }

 List<Contact> contacts = new ArrayList<Contact>();
 if ( root.customer != null && root.customer.contacts != null) {
  for (org.example.entity.Contact jsonContact : root.customer.contacts) {
   Contact contact = Contact.builder()
                 .email(jsonContact.email)
                 .firstName(jsonContact.firstName)
                 .lastName(jsonContact.lastName)
                 .phone(jsonContact.phone)
                 .businessTitle(jsonContact.businessTitle)
                 .build();
   contacts.add(contact);
  }
 }

 Customer customer = Customer.builder()
   .account(account)
   .contacts(contacts)
   .build();

 Contact oportunityTeamContact = null;
 if (root.opportunityTeam != null && root.opportunityTeam.get(0) != null ) {
  oportunityTeamContact = Contact.builder()
                .firstName(root.opportunityTeam.get(0).firstName)
                .lastName(root.opportunityTeam.get(0).lastName)
                .email(root.opportunityTeam.get(0).email)
                .phone(root.opportunityTeam.get(0).phone)
                .businessTitle(root.opportunityTeam.get(0).businessTitle)
                .build();
 }

 List<ExpectedCustomerSpend> expectedCustomerSpends = new
ArrayList<ExpectedCustomerSpend>();
 if ( root.project != null && root.project.expectedCustomerSpend != null) {
  for (org.example.entity.ExpectedCustomerSpend expectedCustomerSpendJson :
root.project.expectedCustomerSpend) {
   ExpectedCustomerSpend expectedCustomerSpend = null;
```

```
    expectedCustomerSpend = ExpectedCustomerSpend.builder()
      .amount(expectedCustomerSpendJson.amount)
      .currencyCode(expectedCustomerSpendJson.currencyCode)
      .frequency(expectedCustomerSpendJson.frequency)
      .targetCompany(expectedCustomerSpendJson.targetCompany)
      .build();
    expectedCustomerSpends.add(expectedCustomerSpend);
   }
        }

        Project project = null;
        if ( root.project != null) {
         project = Project.builder()
                .title(root.project.title)
                .customerBusinessProblem(root.project.customerBusinessProblem)
                .customerUseCase(root.project.customerUseCase)
                .deliveryModels(root.project.deliveryModels)
                .expectedCustomerSpend(expectedCustomerSpends)
                .salesActivities(root.project.salesActivities)
                .competitorName(root.project.competitorName)
                .otherSolutionDescription(root.project.otherSolutionDescription)
                .build();
        }

        SoftwareRevenue softwareRevenue = null;
        if ( root.softwareRevenue != null) {
         MonetaryValue monetaryValue = null;
         if ( root.softwareRevenue.value != null) {
          monetaryValue = MonetaryValue.builder()
             .amount(root.softwareRevenue.value.amount)
             .currencyCode(root.softwareRevenue.value.currencyCode)
             .build();
         }
         softwareRevenue = SoftwareRevenue.builder()
           .deliveryModel(root.softwareRevenue.deliveryModel)
           .effectiveDate(root.softwareRevenue.effectiveDate)
           .expirationDate(root.softwareRevenue.expirationDate)
           .value(monetaryValue)
           .build();
        }

  // Building the Actual CreateOpportunity Request
  CreateOpportunityRequest createOpportunityRequest =
CreateOpportunityRequest.builder()
```

```
    .catalog(CATALOG_TO_USE)
    .clientToken(root.clientToken)
    .primaryNeedsFromAwsWithStrings(root.primaryNeedsFromAws)
    .opportunityType(root.opportunityType)
    .lifeCycle(lifeCycle)
    .marketing(marketing)
    .nationalSecurity(root.nationalSecurity)
    .origin(root.origin)
    .customer(customer)
    .project(project)
    .partnerOpportunityIdentifier(root.partnerOpportunityIdentifier)
    .opportunityTeam(oportunityTeamContact)
    .softwareRevenue(softwareRevenue)
    .build();

  CreateOpportunityResponse response =
 client.createOpportunity(createOpportunityRequest);
  System.out.println("Successfully created: " + response);

  return response;
    }

}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateOpportunity를 참조하세요.

`DisassociateOpportunity`

다음 코드 예시는 `DisassociateOpportunity`의 사용 방법을 보여줍니다.

SDK for Java 2.x

기회 및 관련 엔터티 간의 기존 연결을 제거합니다.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
```

```
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityResponse

/*
Purpose
PC-API -14 Removing a Solution
PC-API -15 Removing an offer
PC-API -16 Removing a product
entity_type = Solutions | AWSProducts | AWSMarketplaceOffers
*/

public class DisassociateOpportunity {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     String entityType = "Solutions";

     String entityIdentifier = "S-0000000";

     DisassociateOpportunityResponse response = getResponse(opportunityId,
 entityType, entityIdentifier );

     ReferenceCodesUtils.formatOutput(response);
    }

 static DisassociateOpportunityResponse getResponse(String opportunityId, String
 entityType, String entityIdentifier) {

  DisassociateOpportunityRequest disassociateOpportunityRequest =
 DisassociateOpportunityRequest.builder()
```

```
    .catalog(Constants.CATALOG_TO_USE)
          .opportunityIdentifier(opportunityId)
          .relatedEntityType(entityType)
          .relatedEntityIdentifier(entityIdentifier)
          .build();

        DisassociateOpportunityResponse response =
 client.disassociateOpportunity(disassociateOpportunityRequest);

        return response;
 }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조에서 DisassociateOpportunity를 참조하세요.

GetAwsOpportunitySummary

다음 코드 예시는 GetAwsOpportunitySummary의 사용 방법을 보여줍니다.

SDK for Java 2.x

AWS 기회의 요약을 검색합니다.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetAwsOpportunitySummaryRequest
import
 software.amazon.awssdk.services.partnercentralselling.model.GetAwsOpportunitySummaryRespons

/*
 * Purpose
```

```
 * PC-API-25 Retrieves a summary of an AWS Opportunity.
 */

public class GetAwsOpportunitySummary {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     GetAwsOpportunitySummaryResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 public static GetAwsOpportunitySummaryResponse getResponse(String opportunityId) {

  GetAwsOpportunitySummaryRequest getOpportunityRequest =
 GetAwsOpportunitySummaryRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .relatedOpportunityIdentifier(opportunityId)
          .build();

  GetAwsOpportunitySummaryResponse response =
 client.getAwsOpportunitySummary(getOpportunityRequest);

        return response;
 }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetAwsOpportunitySummary를 참조하세요.

GetEngagementInvitation

다음 코드 예시는 GetEngagementInvitation의 사용 방법을 보여줍니다.

SDK for Java 2.x

에서 AWS 파트너와 공유하는 참여 초대의 세부 정보를 검색합니다.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationResponse

/*
 * Purpose
 * PC-API-22 Get engagement invitation opportunity
 */

public class GetEngagementInvitation {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     GetEngagementInvitationResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 static GetEngagementInvitationResponse getResponse(String opportunityId) {
```

```
  GetEngagementInvitationRequest getOpportunityRequest =
 GetEngagementInvitationRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
          .build();

  GetEngagementInvitationResponse response =
 client.getEngagementInvitation(getOpportunityRequest);

        return response;
 }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetEngagementInvitation을 참조하세요.

`GetOpportunity`

다음 코드 예시는 `GetOpportunity`의 사용 방법을 보여줍니다.

SDK for Java 2.x

기회를 가져옵니다.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityResponse;

/*
```

```
 * Purpose
 * PC-API-08 Get updated Opportunity
 */

public class GetOpportunity {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     GetOpportunityResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 public static GetOpportunityResponse getResponse(String opportunityId) {

        GetOpportunityRequest getOpportunityRequest =
 GetOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
          .build();

        GetOpportunityResponse response =
 client.getOpportunity(getOpportunityRequest);

        return response;
 }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetOpportunity를 참조하세요.

`ListEngagementInvitations`

다음 코드 예시는 `ListEngagementInvitations`의 사용 방법을 보여줍니다.

SDK for Java 2.x

파트너에게 보낸 참여 초대 목록을 검색합니다.

```
package org.example;

import java.util.ArrayList;
import java.util.List;

import org.example.utils.ReferenceCodesUtils;
import static org.example.utils.Constants.*;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListEngagementInvitationsReques
import
 software.amazon.awssdk.services.partnercentralselling.model.ListEngagementInvitationsRespon
import software.amazon.awssdk.services.partnercentralselling.model.ParticipantType;
import
 software.amazon.awssdk.services.partnercentralselling.model.EngagementInvitationSummary;

public class ListEngagementInvitations {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     List<EngagementInvitationSummary> opportunitySummaries = getResponse();
        ReferenceCodesUtils.formatOutput(opportunitySummaries);
    }

    static List<EngagementInvitationSummary> getResponse() {

  List<EngagementInvitationSummary> opportunitySummaries = new
 ArrayList<EngagementInvitationSummary>();
```

```
  ListEngagementInvitationsRequest listOpportunityRequest =
 ListEngagementInvitationsRequest.builder()
                .catalog(CATALOG_TO_USE)
                .participantType(ParticipantType.RECEIVER)
          .maxResults(5)
          .build();

  ListEngagementInvitationsResponse response =
 client.listEngagementInvitations(listOpportunityRequest);

     opportunitySummaries.addAll(response.engagementInvitationSummaries());

     client.close();

        return opportunitySummaries;
 }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListEngagementInvitations을 참조하세요.

`ListOpportunities`

다음 코드 예시는 `ListOpportunities`의 사용 방법을 보여줍니다.

SDK for Java 2.x

기회를 나열합니다.

```
package org.example;

import java.util.ArrayList;
import java.util.List;

import org.example.utils.ReferenceCodesUtils;
import static org.example.utils.Constants.*;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
```

```
import
 software.amazon.awssdk.services.partnercentralselling.model.ListOpportunitiesRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListOpportunitiesResponse;
import
 software.amazon.awssdk.services.partnercentralselling.model.OpportunitySummary;

/*
 * Purpose
 * PC-API-18 Getting list of Opportunities
 */

public class ListOpportunititesPaging {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {
     List<OpportunitySummary> opportunitySummaries = getResponse();
        ReferenceCodesUtils.formatOutput(opportunitySummaries);
    }

    private static List<OpportunitySummary> getResponse() {
     List<OpportunitySummary> opportunitySummaries = new
 ArrayList<OpportunitySummary>();

  ListOpportunitiesRequest listOpportunityRequest =
 ListOpportunitiesRequest.builder()
                .catalog(CATALOG_TO_USE)
          .maxResults(5)
          .build();

     ListOpportunitiesResponse response =
 client.listOpportunities(listOpportunityRequest);

     opportunitySummaries.addAll(response.opportunitySummaries());

     while (response.nextToken() != null && response.nextToken().length() > 0) {
      listOpportunityRequest = ListOpportunitiesRequest.builder()
                    .catalog(CATALOG_TO_USE)
              .maxResults(5)
```

```
                .nextToken(response.nextToken())
                .build();
       response = client.listOpportunities(listOpportunityRequest);
       opportunitySummaries.addAll(response.opportunitySummaries());
      }

      client.close();

         return opportunitySummaries;
 }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListOpportunities를 참조하세요.

`ListSolutions`

다음 코드 예시는 `ListSolutions`의 사용 방법을 보여줍니다.

SDK for Java 2.x

파트너가 Partner Central에 등록한 파트너 솔루션 목록을 검색합니다.

```
package org.example;

import java.util.ArrayList;
import java.util.List;

import static org.example.utils.Constants.*;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListSolutionsRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListSolutionsResponse;
import software.amazon.awssdk.services.partnercentralselling.model.SolutionBase;

/*
```

```
 * Purpose
 * PC-API-10 Getting list of solutions
 */

public class ListSolutions {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {
     List<SolutionBase> solutionSummaries = getResponse();
        ReferenceCodesUtils.formatOutput(solutionSummaries);
    }

    static List<SolutionBase> getResponse() {
  List<SolutionBase> solutionSummaries = new ArrayList<SolutionBase>();

  ListSolutionsRequest listSolutionsRequest = ListSolutionsRequest.builder()
    .catalog(CATALOG_TO_USE)
          .maxResults(5)
          .build();

     ListSolutionsResponse response = client.listSolutions(listSolutionsRequest);

     solutionSummaries.addAll(response.solutionSummaries());

        return solutionSummaries;
 }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListSolutions를 참조하세요.

`RejectEngagementInvitation`

다음 코드 예시는 `RejectEngagementInvitation`의 사용 방법을 보여줍니다.

SDK for Java 2.x

AWS 공유된 EngagementInvitation을 거부합니다.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.RejectEngagementInvitationReque
import
 software.amazon.awssdk.services.partnercentralselling.model.RejectEngagementInvitationRespo

/*
 * Purpose
 * PC-API-05 AWS Originated(AO) rejection
 */

public class RejectEngagementInvitation {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

  RejectEngagementInvitationResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 static RejectEngagementInvitationResponse getResponse(String invitationId) {

        RejectEngagementInvitationRequest rejectOpportunityRequest =
 RejectEngagementInvitationRequest.builder()
```

```
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(invitationId)
          .rejectionReason("Unable to support")
          .build();

  RejectEngagementInvitationResponse response =
 client.rejectEngagementInvitation(rejectOpportunityRequest);

        return response;
 }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 RejectEngagementInvitation을 참조하세요.

`StartEngagementByAcceptingInvitationTask`

다음 코드 예시는 `StartEngagementByAcceptingInvitationTask`의 사용 방법을 보여줍니다.

SDK for Java 2.x

EngagementInvitation을 수락하여 참여를 시작합니다.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementByAcceptingInvit
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementByAcceptingInvit
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationResponse
```

```
import software.amazon.awssdk.services.partnercentralselling.model.InvitationStatus;

/*
Purpose
PC-API-04: Start Engagement By Accepting InvitationTask for AWS Originated(AO)
 opportunity
*/

public class StartEngagementByAcceptingInvitationTask {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

 static String clientToken = "test-a30d161";

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     StartEngagementByAcceptingInvitationTaskResponse response =
 getResponse(opportunityId);

     if ( response == null) {
      System.out.println("Opportunity is not AWS Originated.");
     } else {
      ReferenceCodesUtils.formatOutput(response);
     }
    }

    private static GetEngagementInvitationResponse getInvitation(String
 invitationId) {

     GetEngagementInvitationRequest getRequest =
 GetEngagementInvitationRequest.builder()
          .catalog(Constants.CATALOG_TO_USE)
          .identifier(invitationId)
          .build();

  GetEngagementInvitationResponse response =
 client.getEngagementInvitation(getRequest);
```

```
        return response;
 }

 static StartEngagementByAcceptingInvitationTaskResponse getResponse(String
 invitationId) {

  if ( getInvitation(invitationId).status().equals(InvitationStatus.PENDING)) {
   StartEngagementByAcceptingInvitationTaskRequest acceptOpportunityRequest =
     StartEngagementByAcceptingInvitationTaskRequest.builder()
     .catalog(Constants.CATALOG_TO_USE)
           .identifier(invitationId)
           .clientToken(clientToken)
           .build();

   StartEngagementByAcceptingInvitationTaskResponse response =
 client.startEngagementByAcceptingInvitationTask(acceptOpportunityRequest);
         return response;
  }
  return null;
 }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartEngagementByAcceptingInvitationTask를 참조하세요.

`StartEngagementFromOpportunityTask`

다음 코드 예시는 `StartEngagementFromOpportunityTask`의 사용 방법을 보여줍니다.

SDK for Java 2.x

참여 초대를 수락하고 파트너의 시스템에서 해당하는 기회를 생성하여 기존 기회에서 참여 프로세스를 시작합니다.

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
```

```
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import software.amazon.awssdk.services.partnercentralselling.model.AwsSubmission;
import
 software.amazon.awssdk.services.partnercentralselling.model.SalesInvolvementType;
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementFromOpportunityT
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementFromOpportunityT
import software.amazon.awssdk.services.partnercentralselling.model.Visibility;

/*
 * Purpose
 * PC-API-01 Partner Originated (PO) opp submission(Start Engagement From
 Opportunity Task for AO Originated Opportunity)
 */

public class StartEngagementFromOpportunityTask {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     StartEngagementFromOpportunityTaskResponse response =
 getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 static StartEngagementFromOpportunityTaskResponse getResponse(String opportunityId)
 {

  StartEngagementFromOpportunityTaskRequest submitOpportunityRequest =
 StartEngagementFromOpportunityTaskRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
```

```
            .clientToken("test-annjqwesdsd99")

 .awsSubmission(AwsSubmission.builder().involvementType(SalesInvolvementType.CO_SELL).visibi
            .build();

  StartEngagementFromOpportunityTaskResponse response =
 client.startEngagementFromOpportunityTask(submitOpportunityRequest);

        return response;
 }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartEngagementFromOpportunityTask를 참조하세요.

`UpdateOpportunity`

다음 코드 예시는 `UpdateOpportunity`의 사용 방법을 보여줍니다.

SDK for Java 2.x

기회를 업데이트합니다.

```
package org.example;

import java.time.Instant;
import java.util.ArrayList;
import java.util.List;

import static org.example.utils.Constants.*;

import org.example.entity.Root;
import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;
import org.example.utils.StringSerializer;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import software.amazon.awssdk.services.partnercentralselling.model.Account;
```

```
import software.amazon.awssdk.services.partnercentralselling.model.Address;
import software.amazon.awssdk.services.partnercentralselling.model.Contact;
import software.amazon.awssdk.services.partnercentralselling.model.Customer;
import
 software.amazon.awssdk.services.partnercentralselling.model.ExpectedCustomerSpend;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityResponse;
import software.amazon.awssdk.services.partnercentralselling.model.LifeCycle;
import software.amazon.awssdk.services.partnercentralselling.model.Marketing;
import software.amazon.awssdk.services.partnercentralselling.model.NextStepsHistory;
import software.amazon.awssdk.services.partnercentralselling.model.Project;
import software.amazon.awssdk.services.partnercentralselling.model.ReviewStatus;
import
 software.amazon.awssdk.services.partnercentralselling.model.UpdateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.UpdateOpportunityResponse;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.ToNumberPolicy;

/*
 * Purpose
 * PC-API-02/06 Update opportunity when LifeCycle.ReviewStatus is not Submitted or
 In-Review
 */

public class UpdateOpportunity {

 static final Gson GSON = new GsonBuilder()
   .setObjectToNumberStrategy(ToNumberPolicy.LAZILY_PARSED_NUMBER)
   .registerTypeAdapter(String.class, new StringSerializer())
   .create();

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

 static String OPPORTUNITY_ORIGIN = ORIGIN_PARTNER_ORIGINATED;
```

```
public static void main(String[] args) {

 String inputFile = "updateOpportunity.json";

 if (args.length > 0)
  inputFile = args[0];

 UpdateOpportunityResponse response = updateOpportunity(inputFile);

 client.close();
}

public static GetOpportunityResponse getResponse(String opportunityId) {

       GetOpportunityRequest getOpportunityRequest =
GetOpportunityRequest.builder()
   .catalog(Constants.CATALOG_TO_USE)
         .identifier(opportunityId)
         .build();

       GetOpportunityResponse response =
client.getOpportunity(getOpportunityRequest);
       System.out.println(opportunityId + ":" + response);
       return response;
}

public static UpdateOpportunityResponse updateOpportunity(String inputFile) {

 String inputString = ReferenceCodesUtils.readInputFileToString(inputFile);

 Root root = GSON.fromJson(inputString, Root.class);
 GetOpportunityResponse response = getResponse(root.identifier);

 if (response != null
   && response.lifeCycle() != null
   && response.lifeCycle().reviewStatus() != null
   && response.lifeCycle().reviewStatus() != ReviewStatus.SUBMITTED
   && response.lifeCycle().reviewStatus() != ReviewStatus.IN_REVIEW) {

  List<NextStepsHistory> nextStepsHistories = new ArrayList<NextStepsHistory>();
  if ( root.lifeCycle != null && root.lifeCycle.nextStepsHistories != null) {
   for (org.example.entity.NextStepsHistory nextStepsHistoryJson :
root.lifeCycle.nextStepsHistories) {
    NextStepsHistory nextStepsHistory = NextStepsHistory.builder()
```

```
        .time(Instant.parse(nextStepsHistoryJson.time))
        .value(nextStepsHistoryJson.value)
                    .build();
      nextStepsHistories.add(nextStepsHistory);
     }
    }

    LifeCycle lifeCycle = null;
    if ( root.lifeCycle != null ) {
     lifeCycle = LifeCycle.builder()
      .closedLostReason(root.lifeCycle.closedLostReason)
      .nextSteps(root.lifeCycle.nextSteps)
      .nextStepsHistory(nextStepsHistories)
      .reviewComments(root.lifeCycle.reviewComments)
      .reviewStatus(root.lifeCycle.reviewStatus)
      .reviewStatusReason(root.lifeCycle.reviewStatusReason)
      .stage(root.lifeCycle.stage)
      .targetCloseDate(root.lifeCycle.targetCloseDate)
      .build();
    }

    Marketing marketing = null;
    if ( root.marketing != null ) {
     marketing = Marketing.builder()
        .awsFundingUsed(root.marketing.awsFundingUsed)
        .campaignName(root.marketing.campaignName)
        .channels(root.marketing.channels)
        .source(root.marketing.source)
        .useCases(root.marketing.useCases)
        .build();

    }

    Address address = null;
    if (root.customer != null && root.customer.account != null &&
  root.customer.account.address != null) {
     address = Address.builder().postalCode(root.customer.account.address.postalCode)
        .stateOrRegion(root.customer.account.address.stateOrRegion)
        .countryCode(root.customer.account.address.countryCode).build();
    }

    Account account = null;
    if (root.customer != null && root.customer.account != null) {
     account = Account.builder().address(address).duns(root.customer.account.duns)
```

```
.industry(root.customer.account.industry).companyName(root.customer.account.companyName)
     .websiteUrl(root.customer.account.websiteUrl).build();
  }

  List<Contact> contacts = new ArrayList<Contact>();
  if ( root.customer != null && root.customer.contacts != null) {
   for (org.example.entity.Contact jsonContact : root.customer.contacts) {
    Contact contact = Contact.builder()
                  .email(jsonContact.email)
                  .firstName(jsonContact.firstName)
                  .lastName(jsonContact.lastName)
                  .phone(jsonContact.phone)
                  .businessTitle(jsonContact.businessTitle)
                  .build();
    contacts.add(contact);
   }
  }

  Customer customer =
Customer.builder().account(account).contacts(contacts).build();

  List<ExpectedCustomerSpend> expectedCustomerSpends = new
ArrayList<ExpectedCustomerSpend>();
  if ( root.project != null && root.project.expectedCustomerSpend != null) {
   for (org.example.entity.ExpectedCustomerSpend expectedCustomerSpendJson :
root.project.expectedCustomerSpend) {
    ExpectedCustomerSpend expectedCustomerSpend = null;
    expectedCustomerSpend = ExpectedCustomerSpend.builder()
      .amount(expectedCustomerSpendJson.amount)
      .currencyCode(expectedCustomerSpendJson.currencyCode)
      .frequency(expectedCustomerSpendJson.frequency)
      .targetCompany(expectedCustomerSpendJson.targetCompany)
      .build();
    expectedCustomerSpends.add(expectedCustomerSpend);
   }
        }

  Project project = null;
  if (root.project != null) {
   project = Project.builder().title(root.project.title)
     .customerBusinessProblem(root.project.customerBusinessProblem)

.customerUseCase(root.project.customerUseCase).deliveryModels(root.project.deliveryModels)
```

```
      .expectedCustomerSpend(expectedCustomerSpends)

 .salesActivities(root.project.salesActivities).competitorName(root.project.competitorName)
      .otherSolutionDescription(root.project.otherSolutionDescription).build();
   }

   // Building the Actual CreateOpportunity Request
   UpdateOpportunityRequest updateOpportunityRequest =
 UpdateOpportunityRequest.builder().catalog(root.catalog)

 .identifier(root.identifier).lastModifiedDate(Instant.parse(root.lastModifiedDate))

 .primaryNeedsFromAwsWithStrings(root.primaryNeedsFromAws).opportunityType(root.opportunityT
     .lifeCycle(lifeCycle)
     .customer(customer)
     .project(project)
     .partnerOpportunityIdentifier(root.partnerOpportunityIdentifier)
     .marketing(marketing)
     .nationalSecurity(root.nationalSecurity)
     .opportunityType(root.opportunityType)
     .build();

   UpdateOpportunityResponse updateResponse =
 client.updateOpportunity(updateOpportunityRequest);
   System.out.println("Successfully updated opportunity: " + updateResponse);

   return updateResponse;
  } else {
   System.out.println("Opportunity cannot be updated.");
   return null;
  }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateOpportunity를 참조하세요.

시나리오

기회에 연결된 엔터티 업데이트

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 이전 엔터티의 연결을 해제합니다.
- 새 엔터티를 연결합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 시나리오 리포지토리에서 전체 예제를 확인하고 설정 및 실행하는 방법을 알아보세요.

기회에 연결된 엔터티 업데이트

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityResponse;
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityResponse

/*
Purpose
PC-API -17 Replacing a solution
*/

public class ReplaceSolution {
```

```
static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
           .region(Region.US_EAST_1)
           .credentialsProvider(DefaultCredentialsProvider.create())
           .httpClient(ApacheHttpClient.builder().build())
           .build();

   public static void main(String[] args) {

    String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

    String entityType = "Solutions";
    String originalEntityIdentifier = "S-0000000";
    String newEntityIdentifier = "S-0011111";

    disassociateOppornititityResponse(opportunityId, entityType,
originalEntityIdentifier );
    AssociateOpportunityResponse associateOpportunityResponse =
associateOpportunityResponse(opportunityId, entityType, newEntityIdentifier );

    ReferenceCodesUtils.formatOutput(associateOpportunityResponse);
   }

private static AssociateOpportunityResponse associateOpportunityResponse(String
opportunityId, String entityType, String entityIdentifier) {

       AssociateOpportunityRequest associateOpportunityRequest =
AssociateOpportunityRequest.builder()
   .catalog(Constants.CATALOG_TO_USE)
         .opportunityIdentifier(opportunityId)
         .relatedEntityType(entityType)
         .relatedEntityIdentifier(entityIdentifier)
         .build();

       AssociateOpportunityResponse response =
client.associateOpportunity(associateOpportunityRequest);

       return response;
}

private static DisassociateOpportunityResponse
disassociateOppornititityResponse(String opportunityId, String entityType, String
entityIdentifier) {
 PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
           .region(Region.US_EAST_1)
```

```
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

        DisassociateOpportunityRequest disassociateOpportunityRequest =
 DisassociateOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .opportunityIdentifier(opportunityId)
          .relatedEntityType(entityType)
          .relatedEntityIdentifier(entityIdentifier)
          .build();

        DisassociateOpportunityResponse response =
 client.disassociateOpportunity(disassociateOpportunityRequest);

        return response;
 }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - AssociateOpportunity
  - DisassociateOpportunity

Java 2.x용 SDK를 사용하는 Amazon Personalize 예제

다음 코드 예제에서는 Personalize와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`CreateBatchInferenceJob`

다음 코드 예시는 `CreateBatchInferenceJob`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        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 "";
        }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateBatchInferenceJob을 참조하세요.

CreateCampaign

다음 코드 예시는 CreateCampaign의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   public static String 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());
           return campaignResponse.campaignArn();
       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return null;
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateCampaign을 참조하세요.

`CreateDataset`

다음 코드 예시는 `CreateDataset`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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 "";
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateDataset를 참조하세요.

`CreateDatasetExportJob`

다음 코드 예시는 `CreateDatasetExportJob`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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 "";
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateDatasetExportJob을 참조하세요.

`CreateDatasetGroup`

다음 코드 예시는 `CreateDatasetGroup`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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 "";
   }
```

도메인 데이터 세트 그룹을 생성합니다.

```
   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 "";
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateDatasetGroup을 참조하세요.

`CreateDatasetImportJob`

다음 코드 예시는 `CreateDatasetImportJob`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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 "";
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateDatasetImportJob을 참조하세요.

`CreateEventTracker`

다음 코드 예시는 `CreateEventTracker`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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;
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateEventTracker를 참조하세요.

`CreateFilter`

다음 코드 예시는 `CreateFilter`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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 "";
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateFilter를 참조하세요.

`CreateRecommender`

다음 코드 예시는 `CreateRecommender`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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 "";
```

```
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateRecommender를 참조하세요.

`CreateSchema`

다음 코드 예시는 `CreateSchema`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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 "";
    }
```

도메인 포함 스키마 생성

```
    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 "";
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateSchema를 참조하세요.

CreateSolution

다음 코드 예시는 CreateSolution의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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 "";
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateSolution을 참조하세요.

CreateSolutionVersion

다음 코드 예시는 CreateSolutionVersion의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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 "";
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateSolutionVersion을 참조하세요.

`DeleteCampaign`

다음 코드 예시는 `DeleteCampaign`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    public static void deleteSpecificCampaign(PersonalizeClient personalizeClient,
String campaignArn) {
        try {
            DeleteCampaignRequest campaignRequest = DeleteCampaignRequest.builder()
                    .campaignArn(campaignArn)
                    .build();

            personalizeClient.deleteCampaign(campaignRequest);
            System.out.println("Delete request sent successfully.");
        } catch (PersonalizeException e) {
            System.err.println("Error deleting campaign: " +
e.awsErrorDetails().errorMessage());
            throw new RuntimeException(e);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteCampaign을 참조하세요.

DeleteEventTracker

다음 코드 예시는 DeleteEventTracker의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteEventTracker를 참조하세요.

DeleteSolution

다음 코드 예시는 DeleteSolution의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteSolution을 참조하세요.

`DescribeCampaign`

다음 코드 예시는 `DescribeCampaign`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeCampaign을 참조하세요.

`DescribeRecipe`

다음 코드 예시는 `DescribeRecipe`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeRecipe를 참조하세요.

`DescribeSolution`

다음 코드 예시는 `DescribeSolution`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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);
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeSolution을 참조하세요.

`ListCampaigns`

다음 코드 예시는 `ListCampaigns`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
```

```
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListCampaigns를 참조하세요.

`ListDatasetGroups`

다음 코드 예시는 `ListDatasetGroups`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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);
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListDatasetGroups를 참조하세요.

`ListRecipes`

다음 코드 예시는 `ListRecipes`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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);
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListRecipes를 참조하세요.

`ListSolutions`

다음 코드 예시는 `ListSolutions`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListSolutions를 참조하세요.

`UpdateCampaign`

다음 코드 예시는 `UpdateCampaign`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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 "";
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateCampaign을 참조하세요.

SDK for Java 2.x를 사용하는 Amazon Personalize Events 예제

다음 코드 예제에서는 Personalize Events와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`PutEvents`

다음 코드 예시는 `PutEvents`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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;
        }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutEvents를 참조하세요.

`PutUsers`

다음 코드 예시는 `PutUsers`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        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;
        }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 PutUsers를 참조하세요.

Java 2.x용 SDK를 사용하는 Amazon Personalize 런타임 예제

다음 코드 예제에서는 Personalize 런타임과 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`GetPersonalizedRanking`

다음 코드 예시는 `GetPersonalizedRanking`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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;
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 GetPersonalizedRanking을 참조하세요.

`GetRecommendations`

다음 코드 예시는 `GetRecommendations`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

권장 품목 목록을 확인하세요.

```
   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);
        }
    }
```

도메인 데이터세트 그룹에 생성된 추천에서 추천한 권장 품목 목록 가져오기.

```
    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);
        }
    }
```

추천을 요청할 때는 필터를 사용하세요.

```
   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);
       }
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 GetRecommendations를 참조하세요.

SDK for Java 2.x를 사용하는 Amazon Pinpoint 예제

다음 코드 예제에서는 Amazon Pinpoint와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`CreateApp`

다음 코드 예시는 `CreateApp`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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 "";
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateApp을 참조하세요.

`CreateCampaign`

다음 코드 예시는 `CreateCampaign`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

캠페인을 생성합니다.

```
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;
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateCampaign을 참조하세요.

`CreateExportJob`

다음 코드 예시는 `CreateExportJob`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

엔드포인트를 내보내세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateExportJob을 참조하세요.

`CreateImportJob`

다음 코드 예시는 `CreateImportJob`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

세그먼트를 가져오세요.

```
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;
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateImportJob을 참조하세요.

`CreateSegment`

다음 코드 예시는 `CreateSegment`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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;
        }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateSegment를 참조하세요.

`DeleteApp`

다음 코드 예시는 `DeleteApp`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

애플리케이션을 삭제합니다.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteApp을 참조하세요.

`DeleteEndpoint`

다음 코드 예시는 `DeleteEndpoint`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

엔드포인트를 삭제합니다.

```
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");
    }
```

```
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteEndpoint를 참조하세요.

`GetEndpoint`

다음 코드 예시는 `GetEndpoint`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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");
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 GetEndpoint를 참조하세요.

`GetSegments`

다음 코드 예시는 `GetSegments`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

세그먼트를 나열하세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 GetSegments를 참조하세요.

`GetSmsChannel`

다음 코드 예시는 `GetSmsChannel`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 GetSmsChannel을 참조하세요.

`GetUserEndpoints`

다음 코드 예시는 `GetUserEndpoints`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 GetUserEndpoints를 참조하세요.

`SendMessages`

다음 코드 예시는 SendMessages의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

이메일 메시지를 전송합니다.

```
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);
        }
    }
}
```

CC 값을 포함하여 이메일 메시지를 전송합니다.

```
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();
        }
    }
}
```

SMS 메시지를 전송합니다.

```
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);
                }
        }
}
```

SMS 메시지를 일괄 전송합니다.

```
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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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);
        }
```

```
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 SendMessages를 참조하세요.

`UpdateEndpoint`

다음 코드 예시는 `UpdateEndpoint`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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;
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateEndpoint를 참조하세요.

Java 2.x용 SDK를 사용하는 Amazon Pinpoint SMS 및 음성 API 예제

다음 코드 예제에서는 Amazon Pinpoint SMS 및 음성 API와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`SendVoiceMessage`

다음 코드 예시는 `SendVoiceMessage`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 SendVoiceMessage를 참조하세요.

Java 2.x용 SDK를 사용하는 Amazon Polly 예제

다음 코드 예제에서는 AWS SDK for Java 2.x Amazon Polly에서를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`DescribeVoices`

다음 코드 예시는 `DescribeVoices`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeVoices를 참조하세요.

`ListLexicons`

다음 코드 예시는 `ListLexicons`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListLexicons를 참조하세요.

`SynthesizeSpeech`

다음 코드 예시는 `SynthesizeSpeech`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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;
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 SynthesizeSpeech를 참조하세요.

시나리오

고객 피드백 분석을 위한 애플리케이션 생성

다음 코드 예제에서는 고객 의견 카드를 분석하고, 원어에서 번역하고, 감정을 파악하고, 번역된 텍스트에서 오디오 파일을 생성하는 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

이 예제 애플리케이션은 고객 피드백 카드를 분석하고 저장합니다. 특히 뉴욕시에 있는 가상 호텔의 필요를 충족합니다. 호텔은 다양한 언어의 고객들로부터 물리적인 의견 카드의 형태로 피드백을 받습니다. 피드백은 웹 클라이언트를 통해 앱에 업로드됩니다. 의견 카드의 이미지가 업로드된 후 다음 단계가 수행됩니다.

- Amazon Textract를 사용하여 이미지에서 텍스트가 추출됩니다.
- Amazon Comprehend가 추출된 텍스트와 해당 언어의 감정을 파악합니다.
- 추출된 텍스트는 Amazon Translate를 사용하여 영어로 번역됩니다.
- Amazon Polly가 추출된 텍스트에서 오디오 파일을 합성합니다.

전체 앱은 AWS CDK를 사용하여 배포할 수 있습니다. 소스 코드와 배포 지침은 GitHub의 프로젝트를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

Java 2.x용 SDK를 사용하는 Amazon RDS 예제

다음 코드 예제에서는 AWS SDK for Java 2.x Amazon RDS에서를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시작하기

Hello Amazon RDS

다음 코드 예제에서는 Amazon RDS 사용을 시작하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeDBInstances를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 사용자 지정 DB 파라미터 그룹을 생성하고 파라미터 값을 설정합니다.

- 파라미터 그룹을 사용하도록 구성된 DB 인스턴스를 생성합니다. DB 인스턴스에는 데이터베이스도 포함되어 있습니다.
- 인스턴스의 스냅샷을 만듭니다.
- 인스턴스 및 파라미터 그룹을 삭제합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

여러 작업을 실행합니다.

```
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)
            .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 userName,
        String userPassword) {

    try {
        CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .allocatedStorage(100)
            .dbName(dbName)
            .engine("mysql")
            .dbInstanceClass("db.t3.medium") // Updated to a supported class
            .engineVersion("8.0.32")         // Updated to a supported version
            .storageType("gp2")             // Changed to General Purpose SSD
(gp2)
            .masterUsername(userName)
            .masterUserPassword(userPassword)
            .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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance

- DeleteDBParameterGroup
- DescribeDBEngineVersions
- DescribeDBInstances
- DescribeDBParameterGroups
- DescribeDBParameters
- DescribeDBSnapshots
- DescribeOrderableDBInstanceOptions
- ModifyDBParameterGroup

작업

`CreateDBInstance`

다음 코드 예시는 `CreateDBInstance`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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.t3.medium") // Updated to a supported class
                .engineVersion("8.0.32")         // Updated to a supported version
                .storageType("gp2")             // Changed to General Purpose SSD
(gp2)
                .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);
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateDBInstance를 참조하세요.

`CreateDBParameterGroup`

다음 코드 예시는 `CreateDBParameterGroup`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateDBParameterGroup을 참조하세요.

`CreateDBSnapshot`

다음 코드 예시는 `CreateDBSnapshot`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateDBSnapshot을 참조하세요.

`DeleteDBInstance`

다음 코드 예시는 `DeleteDBInstance`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteDBInstance를 참조하세요.

`DeleteDBParameterGroup`

다음 코드 예시는 `DeleteDBParameterGroup`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteDBParameterGroup을 참조하세요.

`DescribeAccountAttributes`

다음 코드 예시는 `DescribeAccountAttributes`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeAccountAttributes를 참조하세요.

`DescribeDBEngineVersions`

다음 코드 예시는 `DescribeDBEngineVersions`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeDBEngineVersions을 참조하세요.

`DescribeDBInstances`

다음 코드 예시는 `DescribeDBInstances`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeDBInstances를 참조하세요.

`DescribeDBParameterGroups`

다음 코드 예시는 `DescribeDBParameterGroups`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeDBParameterGroups를 참조하세요.

`DescribeDBParameters`

다음 코드 예시는 `DescribeDBParameters`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeDBParameters 참조하세요.

GenerateRDSAuthToken

다음 코드 예시는 GenerateRDSAuthToken의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

RDSUtilities 클래스를 사용하여 인증 토큰을 생성합니다.

```
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 "";
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GenerateRDSAuthToken을 참조하세요.

`ModifyDBInstance`

다음 코드 예시는 `ModifyDBInstance`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ModifyDBInstance를 참조하세요.

`ModifyDBParameterGroup`

다음 코드 예시는 `ModifyDBParameterGroup`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ModifyDBParameterGroup을 참조하세요.

`RebootDBInstance`

다음 코드 예시는 `RebootDBInstance`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
```

```
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 RebootDBInstance를 참조하세요.

시나리오

Aurora 서버리스 작업 항목 트래커 만들기

다음 코드 예제에서는 Amazon Aurora Serverless 데이터베이스에서 작업 항목을 추적하고 Amazon Simple Email Service(Amazon SES)를 사용하여 보고서를 보내는 웹 애플리케이션을 만드는 방법을 보여줍니다.

SDK for Java 2.x

Amazon RDS 데이터베이스에 저장된 작업 항목을 추적하고 보고하는 웹 애플리케이션을 만드는 방법을 보여줍니다.

Amazon Aurora Serverless 데이터를 쿼리하고 React 애플리케이션에서 사용하도록 Spring REST API를 설정하는 방법에 대한 지침과 전체 소스 코드는 GitHub에서 전체 예제를 참조하세요.

JDBC API를 사용한 예제를 설정하고 실행하는 방법에 대한 전체 소스 코드와 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Aurora
- Amazon RDS
- Amazon RDS 데이터 서비스
- Amazon SES

서버리스 예제

Lambda 함수를 사용하여 Amazon RDS 데이터베이스에 연결

다음 코드 예제는 RDS 데이터베이스에 연결하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 간단한 데이터베이스 요청을 하고 결과를 반환합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Java를 사용하여 Lambda 함수에서 Amazon RDS 데이터베이스에 연결

```
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();
    }
}
```

SDK for Java 2.x를 사용한 Amazon RDS 데이터 서비스 예제

다음 코드 예제에서는 Amazon RDS Data Service와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시나리오

Aurora 서버리스 작업 항목 트래커 만들기

다음 코드 예제에서는 Amazon Aurora Serverless 데이터베이스에서 작업 항목을 추적하고 Amazon Simple Email Service(Amazon SES)를 사용하여 보고서를 보내는 웹 애플리케이션을 만드는 방법을 보여줍니다.

SDK for Java 2.x

Amazon RDS 데이터베이스에 저장된 작업 항목을 추적하고 보고하는 웹 애플리케이션을 만드는 방법을 보여줍니다.

Amazon Aurora Serverless 데이터를 쿼리하고 React 애플리케이션에서 사용하도록 Spring REST API를 설정하는 방법에 대한 지침과 전체 소스 코드는 GitHub에서 전체 예제를 참조하세요.

JDBC API를 사용한 예제를 설정하고 실행하는 방법에 대한 전체 소스 코드와 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Aurora
- Amazon RDS
- Amazon RDS 데이터 서비스
- Amazon SES

Java 2.x용 SDK를 사용하는 Amazon Redshift 예제

다음 코드 예제에서는 AWS SDK for Java 2.x Amazon Redshift에서를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시작하기

Hello Amazon Redshift

다음 코드 예시에서는 Amazon Redshift 사용을 시작하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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()));
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeClusters를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- Redshift 클러스터를 생성합니다.
- 클러스터의 데이터베이스를 나열합니다.
- Movies라는 테이블을 생성합니다.
- Movies 테이블을 채웁니다.
- Movies 테이블을 연도별로 쿼리합니다.
- Redshift 클러스터를 수정합니다.
- Amazon Redshift 클러스터를 삭제합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon Redshift 기능을 시연하는 대화형 시나리오를 실행합니다.

```
import com.example.redshift.User;
import com.google.gson.Gson;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.redshift.model.ClusterAlreadyExistsException;
import software.amazon.awssdk.services.redshift.model.CreateClusterResponse;
import software.amazon.awssdk.services.redshift.model.DeleteClusterResponse;
import software.amazon.awssdk.services.redshift.model.ModifyClusterResponse;
import software.amazon.awssdk.services.redshift.model.RedshiftException;
import software.amazon.awssdk.services.redshiftdata.model.ExecuteStatementResponse;
import software.amazon.awssdk.services.redshiftdata.model.RedshiftDataException;
```

```
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including 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 that specifies user name and password, 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. 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", "-");
    private static final Logger logger =
 LoggerFactory.getLogger(RedshiftScenario.class);
```

```
   static RedshiftActions redshiftActions = new RedshiftActions();
   public static void main(String[] args) throws Exception {
       final String usage = """

           Usage:
               <jsonFilePath> <secretName>\s

           Where:
               jsonFilePath - The path to the Movies JSON file (you can locate that
file in ../../../resources/sample_files/movies.json)
               secretName - The name of the secret that belongs to Secret Manager
that stores the user name and password used in this scenario.
           """;

       if (args.length != 2) {
           logger.info(usage);
           return;
       }

       String jsonFilePath = args[0];
       String secretName = args[1];
       Scanner scanner = new Scanner(System.in);
       logger.info(DASHES);
       logger.info("Welcome to the Amazon Redshift SDK Basics scenario.");
       logger.info("""
           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.
           """);

       logger.info("Lets get started...");
       logger.info("""
           First, we will retrieve the user name and password from Secrets Manager.
```

```
            Using Amazon Secrets Manager to store Redshift credentials provides
 several security benefits.
            It allows you to securely store and manage sensitive information, such
 as passwords, API keys, and
            database credentials, without embedding them directly in your
 application code.

            More information can be found here:

            https://docs.aws.amazon.com/secretsmanager/latest/userguide/
integrating_how-services-use-secrets_RS.html
            """);
        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
 User.class);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        try {
            runScenario(user, scanner, jsonFilePath);
        } catch (RuntimeException e) {
            e.printStackTrace();
        } catch (Throwable e) {
            throw new RuntimeException(e);
        }
    }

    private static void runScenario(User user, Scanner scanner,  String
 jsonFilePath) throws Throwable {
        String databaseName = "dev";
        System.out.println(DASHES);
        logger.info("Create a Redshift Cluster");
        logger.info("A Redshift cluster refers to the collection of computing
 resources and storage that work together to process and analyze large volumes of
 data.");
        logger.info("Enter a cluster id value or accept the default by hitting Enter
 (default is redshift-cluster-movies): ");
        String userClusterId = scanner.nextLine();
        String clusterId = userClusterId.isEmpty() ? "redshift-cluster-movies" :
 userClusterId;
        try {
            CompletableFuture<CreateClusterResponse> future =
 redshiftActions.createClusterAsync(clusterId, user.getUserName(),
 user.getUserPassword());
```

```
            CreateClusterResponse response = future.join();
            logger.info("Cluster successfully created. Cluster Identifier {} ",
 response.cluster().clusterIdentifier());

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof ClusterAlreadyExistsException) {
                logger.info("The Cluster {} already exists. Moving on...",
 clusterId);
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Wait until {} is available.", clusterId);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
 redshiftActions.waitForClusterReadyAsync(clusterId);
            future.join();
            logger.info("Cluster is ready!");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftException redshiftEx) {
                logger.info("Redshift error occurred: Error message: {}, Error code
 {}", redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(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);

        logger.info(databaseInfo);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("List databases in {} ",clusterId);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
redshiftActions.listAllDatabasesAsync(clusterId, user.getUserName(), "dev");
            future.join();
            logger.info("Databases listed successfully.");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.error("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.error("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Now you will create a table named Movies.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<ExecuteStatementResponse> future =
redshiftActions.createTableAsync(clusterId, databaseName, user.getUserName());
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
```

```
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Populate the Movies table using the Movies.json file.");
        logger.info("Specify the number of records you would like to add to the
Movies Table.");
        logger.info("Please enter a value between 50 and 200.");
        int numRecords;
        do {
            logger.info("Enter a value: ");
            while (!scanner.hasNextInt()) {
                logger.info("Invalid input. Please enter a value between 50 and
200.");
                logger.info("Enter a year: ");
                scanner.next();
            }
            numRecords = scanner.nextInt();
        } while (numRecords < 50 || numRecords > 200);
        try {
            redshiftActions.popTableAsync(clusterId, databaseName,
user.getUserName(), jsonFilePath, numRecords).join();  // Wait for the operation to
complete
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Query the Movies table by year. Enter a value between
2012-2014.");
        int movieYear;
        do {
            logger.info("Enter a year: ");
            while (!scanner.hasNextInt()) {
```

```
                logger.info("Invalid input. Please enter a valid year between 2012
and 2014.");
                logger.info("Enter a year: ");
                scanner.next();
            }
            movieYear = scanner.nextInt();
            scanner.nextLine();
        } while (movieYear < 2012 || movieYear > 2014);

        String id;
        try {
            CompletableFuture<String> future =
redshiftActions.queryMoviesByYearAsync(databaseName, user.getUserName(), movieYear,
clusterId);
            id = future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        logger.info("The identifier of the statement is " + id);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
redshiftActions.checkStatementAsync(id);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
```

```
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future = redshiftActions.getResultsAsync(id);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Now you will modify the Redshift cluster.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<ModifyClusterResponse> future =
redshiftActions.modifyClusterAsync(clusterId);;
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Would you like to delete the Amazon Redshift cluster? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete {} ", clusterId);
```

```
            waitForInputToContinue(scanner);
            try {
                CompletableFuture<DeleteClusterResponse> future =
redshiftActions.deleteRedshiftClusterAsync(clusterId);;
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof RedshiftDataException redshiftEx) {
                    logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: {}",
rt.getMessage());
                }
                throw cause;
            }
        } else {
            logger.info("The {}  was not deleted", clusterId);
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("This concludes the Amazon Redshift SDK Basics scenario.");
        logger.info(DASHES);
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_EAST_1;
        return SecretsManagerClient.builder()
            .region(region)
            .build();
    }

    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.");
            }
        }
    }

    // Get the Amazon Redshift credentials from AWS Secrets Manager.
    private static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
            .secretId(secretName)
            .build();

        GetSecretValueResponse valueResponse =
 secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }
}
```

Amazon Redshift SDK 메서드의 래퍼 클래스입니다.

```
public class RedshiftActions {

    private static final Logger logger =
 LoggerFactory.getLogger(RedshiftActions.class);
    private static RedshiftDataAsyncClient redshiftDataAsyncClient;

    private static RedshiftAsyncClient redshiftAsyncClient;

    private static RedshiftAsyncClient getAsyncClient() {
        if (redshiftAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
```

```
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            redshiftAsyncClient = RedshiftAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return redshiftAsyncClient;
    }

    private static RedshiftDataAsyncClient getAsyncDataClient() {
        if (redshiftDataAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            redshiftDataAsyncClient = RedshiftDataAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return redshiftDataAsyncClient;
    }

    /**
     * Creates a new Amazon Redshift cluster asynchronously.
     * @param clusterId     the unique identifier for the cluster
     * @param username      the username for the administrative user
     * @param userPassword  the password for the administrative user
     * @return a CompletableFuture that represents the asynchronous operation of
creating the cluster
     * @throws RuntimeException if the cluster creation fails
```

```
     */
    public CompletableFuture<CreateClusterResponse> createClusterAsync(String
clusterId, String username, String userPassword) {
        CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .masterUsername(username)
            .masterUserPassword(userPassword)
            .nodeType("ra3.4xlarge")
            .publiclyAccessible(true)
            .numberOfNodes(2)
            .build();

        return getAsyncClient().createCluster(clusterRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Created cluster ");
                } else {
                    throw new RuntimeException("Failed to create cluster: " +
exception.getMessage(), exception);
                }
            });
    }

    /**
     * Waits asynchronously for the specified cluster to become available.
     * @param clusterId the identifier of the cluster to wait for
     * @return a {@link CompletableFuture} that completes when the cluster is ready
     */
    public CompletableFuture<Void> waitForClusterReadyAsync(String clusterId) {
        DescribeClustersRequest clustersRequest = DescribeClustersRequest.builder()
            .clusterIdentifier(clusterId)
            .build();

        logger.info("Waiting for cluster to become available. This may take a few
minutes.");
        long startTime = System.currentTimeMillis();

        // Recursive method to poll the cluster status.
        return checkClusterStatusAsync(clustersRequest, startTime);
    }

    private CompletableFuture<Void> checkClusterStatusAsync(DescribeClustersRequest
clustersRequest, long startTime) {
        return getAsyncClient().describeClusters(clustersRequest)
```

```
            .thenCompose(clusterResponse -> {
                List<Cluster> clusterList = clusterResponse.clusters();
                boolean clusterReady = false;
                for (Cluster cluster : clusterList) {
                    if ("available".equals(cluster.clusterStatus())) {
                        clusterReady = true;
                        break;
                    }
                }

                if (clusterReady) {
                    logger.info(String.format("Cluster is available!"));
                    return CompletableFuture.completedFuture(null);
                } else {
                    long elapsedTimeMillis = System.currentTimeMillis() - startTime;
                    long elapsedSeconds = elapsedTimeMillis / 1000;
                    long minutes = elapsedSeconds / 60;
                    long seconds = elapsedSeconds % 60;
                    System.out.printf("\rElapsed Time: %02d:%02d - Waiting for
cluster...", minutes, seconds);
                    System.out.flush();

                    // Wait 1 second before the next status check
                    return CompletableFuture.runAsync(() -> {
                        try {
                            TimeUnit.SECONDS.sleep(1);
                        } catch (InterruptedException e) {
                            throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
                        }
                    }).thenCompose(ignored ->
checkClusterStatusAsync(clustersRequest, startTime));
                }
            }).exceptionally(exception -> {
                throw new RuntimeException("Failed to get cluster status: " +
exception.getMessage(), exception);
            });
    }

    /**
     * Lists all databases asynchronously for the specified cluster, database user,
and database.
     * @param clusterId the identifier of the cluster to list databases for
     * @param dbUser the database user to use for the list databases request
```

```
     * @param database the database to list databases for
     * @return a {@link CompletableFuture} that completes when the database listing
is complete, or throws a {@link RuntimeException} if there was an error
     */
    public CompletableFuture<Void> listAllDatabasesAsync(String clusterId, String
dbUser, String database) {
        ListDatabasesRequest databasesRequest = ListDatabasesRequest.builder()
            .clusterIdentifier(clusterId)
            .dbUser(dbUser)
            .database(database)
            .build();

        // Asynchronous paginator for listing databases.
        ListDatabasesPublisher databasesPaginator =
getAsyncDataClient().listDatabasesPaginator(databasesRequest);
        CompletableFuture<Void> future = databasesPaginator.subscribe(response -> {
            response.databases().forEach(db -> {
                logger.info("The database name is {} ", db);
            });
        });

        // Return the future for asynchronous handling.
        return future.exceptionally(exception -> {
            throw new RuntimeException("Failed to list databases: " +
exception.getMessage(), exception);
        });
    }

    /**
     * Creates an asynchronous task to execute a SQL statement for creating a new
table.
     *
     * @param clusterId    the identifier of the Amazon Redshift cluster
     * @param databaseName the name of the database to create the table in
     * @param userName     the username to use for the database connection
     * @return a {@link CompletableFuture} that completes with the result of the SQL
statement execution
     * @throws RuntimeException if there is an error creating the table
     */
    public CompletableFuture<ExecuteStatementResponse> createTableAsync(String
clusterId, String databaseName, String userName) {
        ExecuteStatementRequest createTableRequest =
ExecuteStatementRequest.builder()
            .clusterIdentifier(clusterId)
```

```
            .dbUser(userName)
            .database(databaseName)
            .sql("CREATE TABLE Movies (" +
                "id INT PRIMARY KEY, " +
                "title VARCHAR(100), " +
                "year INT)")
            .build();

        return getAsyncDataClient().executeStatement(createTableRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Error creating table: " +
exception.getMessage(), exception);
                } else {
                    logger.info("Table created: Movies");
                }
            });
    }

    /**
     * Asynchronously pops a table from a JSON file.
     *
     * @param clusterId   the ID of the cluster
     * @param databaseName the name of the database
     * @param userName    the username
     * @param fileName    the name of the JSON file
     * @param number      the number of records to process
     * @return a CompletableFuture that completes with the number of records added
to the Movies table
     */
    public CompletableFuture<Integer> popTableAsync(String clusterId, String
databaseName, String userName, String fileName, int number) {
        return CompletableFuture.supplyAsync(() -> {
                try {
                    JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                    JsonNode rootNode = new ObjectMapper().readTree(parser);
                    Iterator<JsonNode> iter = rootNode.iterator();
                    return iter;
                } catch (IOException e) {
                    throw new RuntimeException("Failed to read or parse JSON file: "
+ e.getMessage(), e);
                }
```

```
            }).thenCompose(iter -> processNodesAsync(clusterId, databaseName,
userName, iter, number))
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    logger.info("Error {} ", exception.getMessage());
                } else {
                    logger.info("{} records were added to the Movies table." ,
result);
                }
            });
   }

   private CompletableFuture<Integer> processNodesAsync(String clusterId, String
databaseName, String userName, Iterator<JsonNode> iter, int number) {
       return CompletableFuture.supplyAsync(() -> {
           int t = 0;
           try {
               while (iter.hasNext()) {
                   if (t == number)
                       break;
                   JsonNode currentNode = 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);";
                   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);

                    ExecuteStatementRequest insertStatementRequest =
ExecuteStatementRequest.builder()
                        .clusterIdentifier(clusterId)
                        .sql(sqlStatement)
                        .database(databaseName)
                        .dbUser(userName)
                        .parameters(parameterList)
                        .build();

                    getAsyncDataClient().executeStatement(insertStatementRequest);
                    logger.info("Inserted: " + title + " (" + year + ")");
                    t++;
                }
            } catch (RedshiftDataException e) {
                throw new RuntimeException("Error inserting data: " +
e.getMessage(), e);
            }
            return t;
        });
    }

    /**
     * Checks the status of an SQL statement asynchronously and handles the
completion of the statement.
     *
     * @param sqlId the ID of the SQL statement to check
     * @return a {@link CompletableFuture} that completes when the SQL statement's
status is either "FINISHED" or "FAILED"
     */
    public CompletableFuture<Void> checkStatementAsync(String sqlId) {
        DescribeStatementRequest statementRequest =
DescribeStatementRequest.builder()
            .id(sqlId)
            .build();

        return getAsyncDataClient().describeStatement(statementRequest)
            .thenCompose(response -> {
                String status = response.statusAsString();
                logger.info("... Status: {} ", status);

                if ("FAILED".equals(status)) {
                    throw new RuntimeException("The Query Failed. Ending program");
```

```
                } else if ("FINISHED".equals(status)) {
                    return CompletableFuture.completedFuture(null);
                } else {
                    // Sleep for 1 second and recheck status
                    return CompletableFuture.runAsync(() -> {
                        try {
                            TimeUnit.SECONDS.sleep(1);
                        } catch (InterruptedException e) {
                            throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
                        }
                    }).thenCompose(ignore -> checkStatementAsync(sqlId)); //
Recursively call until status is FINISHED or FAILED
                }
            }).whenComplete((result, exception) -> {
                if (exception != null) {
                    // Handle exceptions
                    logger.info("Error: {} ", exception.getMessage());
                } else {
                    logger.info("The statement is finished!");
                }
            });
    }

    /**
     * Asynchronously retrieves the results of a statement execution.
     *
     * @param statementId the ID of the statement for which to retrieve the results
     * @return a {@link CompletableFuture} that completes when the statement result
has been processed
     */
    public CompletableFuture<Void> getResultsAsync(String statementId) {
        GetStatementResultRequest resultRequest =
GetStatementResultRequest.builder()
            .id(statementId)
            .build();

        return getAsyncDataClient().getStatementResult(resultRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    logger.info("Error getting statement result {} ",
exception.getMessage());
                    throw new RuntimeException("Error getting statement result: " +
exception.getMessage(), exception);
```

```
                }

                // Extract and print the field values using streams if the response
is valid.
                response.records().stream()
                    .flatMap(List::stream)
                    .map(Field::stringValue)
                    .filter(value -> value != null)
                    .forEach(value -> System.out.println("The Movie title field is "
+ value));

                return response;
            }).thenAccept(response -> {
                // Optionally add more logic here if needed after handling the
response
            });
    }


    /**
     * Asynchronously queries movies by a given year from a Redshift database.
     *
     * @param database    the name of the database to query
     * @param dbUser      the user to connect to the database with
     * @param year        the year to filter the movies by
     * @param clusterId   the identifier of the Redshift cluster to connect to
     * @return a {@link CompletableFuture} containing the response ID of the
executed SQL statement
     */
    public CompletableFuture<String> queryMoviesByYearAsync(String database,
                                                            String dbUser,
                                                            int year,
                                                            String clusterId)
{

        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();

        return CompletableFuture.supplyAsync(() -> {
            try {
                ExecuteStatementResponse response =
getAsyncDataClient().executeStatement(statementRequest).join(); // Use join() to
wait for the result
                return response.id();
            } catch (RedshiftDataException e) {
                throw new RuntimeException("Error executing statement: " +
e.getMessage(), e);
            }
        }).exceptionally(exception -> {
            logger.info("Error: {}", exception.getMessage());
            return "";
        });
    }

    /**
     * Modifies an Amazon Redshift cluster asynchronously.
     *
     * @param clusterId the identifier of the cluster to be modified
     * @return a {@link CompletableFuture} that completes when the cluster
modification is complete
     */
    public CompletableFuture<ModifyClusterResponse> modifyClusterAsync(String
clusterId) {
        ModifyClusterRequest modifyClusterRequest = ModifyClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .preferredMaintenanceWindow("wed:07:30-wed:08:00")
            .build();

        return getAsyncClient().modifyCluster(modifyClusterRequest)
            .whenComplete((clusterResponse, exception) -> {
                if (exception != null) {
                    if (exception.getCause() instanceof RedshiftException) {
                        logger.info("Error: {} ", exception.getMessage());
                    } else {
                        logger.info("Unexpected error: {} ",
exception.getMessage());
                    }
```

```
                } else {
                    logger.info("The modified cluster was successfully modified and
 has "
                        + clusterResponse.cluster().preferredMaintenanceWindow() + "
 as the maintenance window");
                }
            });
    }

    /**
     * Deletes a Redshift cluster asynchronously.
     *
     * @param clusterId the identifier of the Redshift cluster to be deleted
     * @return a {@link CompletableFuture} that represents the asynchronous
 operation of deleting the Redshift cluster
     */
    public CompletableFuture<DeleteClusterResponse>
 deleteRedshiftClusterAsync(String clusterId) {
        DeleteClusterRequest deleteClusterRequest = DeleteClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .skipFinalClusterSnapshot(true)
            .build();

        return getAsyncClient().deleteCluster(deleteClusterRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    // Handle exceptions
                    if (exception.getCause() instanceof RedshiftException) {
                        logger.info("Error: {}", exception.getMessage());
                    } else {
                        logger.info("Unexpected error: {}", exception.getMessage());
                    }
                } else {
                    // Handle successful response
                    logger.info("The status is {}",
 response.cluster().clusterStatus());
                }
            });
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.

- CreateCluster
- DescribeClusters
- DescribeStatement
- ExecuteStatement
- GetStatementResult
- ListDatabasesPaginator
- ModifyCluster

작업

`CreateCluster`

다음 코드 예시는 `CreateCluster`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

클러스터를 생성합니다.

```
   /**
    * Creates a new Amazon Redshift cluster asynchronously.
    * @param clusterId     the unique identifier for the cluster
    * @param username      the username for the administrative user
    * @param userPassword  the password for the administrative user
    * @return a CompletableFuture that represents the asynchronous operation of
creating the cluster
    * @throws RuntimeException if the cluster creation fails
    */
   public CompletableFuture<CreateClusterResponse> createClusterAsync(String
clusterId, String username, String userPassword) {
       CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
           .clusterIdentifier(clusterId)
           .masterUsername(username)
           .masterUserPassword(userPassword)
```

```
            .nodeType("ra3.4xlarge")
            .publiclyAccessible(true)
            .numberOfNodes(2)
            .build();

        return getAsyncClient().createCluster(clusterRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Created cluster ");
                } else {
                    throw new RuntimeException("Failed to create cluster: " +
exception.getMessage(), exception);
                }
            });
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateCluster를 참조하세요.

`DeleteCluster`

다음 코드 예시는 `DeleteCluster`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

클러스터를 삭제합니다.

```
    /**
     * Deletes a Redshift cluster asynchronously.
     *
     * @param clusterId the identifier of the Redshift cluster to be deleted
     * @return a {@link CompletableFuture} that represents the asynchronous
operation of deleting the Redshift cluster
     */
    public CompletableFuture<DeleteClusterResponse>
deleteRedshiftClusterAsync(String clusterId) {
```

```
        DeleteClusterRequest deleteClusterRequest = DeleteClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .skipFinalClusterSnapshot(true)
            .build();

        return getAsyncClient().deleteCluster(deleteClusterRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    // Handle exceptions
                    if (exception.getCause() instanceof RedshiftException) {
                        logger.info("Error: {}", exception.getMessage());
                    } else {
                        logger.info("Unexpected error: {}", exception.getMessage());
                    }
                } else {
                    // Handle successful response
                    logger.info("The status is {}",
response.cluster().clusterStatus());
                }
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteCluster를 참조하세요.

`DescribeClusters`

다음 코드 예시는 `DescribeClusters`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

클러스터를 설명하세요.

```
    /**
     * Waits asynchronously for the specified cluster to become available.
     * @param clusterId the identifier of the cluster to wait for
```

```
     * @return a {@link CompletableFuture} that completes when the cluster is ready
     */
    public CompletableFuture<Void> waitForClusterReadyAsync(String clusterId) {
        DescribeClustersRequest clustersRequest = DescribeClustersRequest.builder()
            .clusterIdentifier(clusterId)
            .build();

        logger.info("Waiting for cluster to become available. This may take a few
minutes.");
        long startTime = System.currentTimeMillis();

        // Recursive method to poll the cluster status.
        return checkClusterStatusAsync(clustersRequest, startTime);
    }

    private CompletableFuture<Void> checkClusterStatusAsync(DescribeClustersRequest
clustersRequest, long startTime) {
        return getAsyncClient().describeClusters(clustersRequest)
            .thenCompose(clusterResponse -> {
                List<Cluster> clusterList = clusterResponse.clusters();
                boolean clusterReady = false;
                for (Cluster cluster : clusterList) {
                    if ("available".equals(cluster.clusterStatus())) {
                        clusterReady = true;
                        break;
                    }
                }

                if (clusterReady) {
                    logger.info(String.format("Cluster is available!"));
                    return CompletableFuture.completedFuture(null);
                } else {
                    long elapsedTimeMillis = System.currentTimeMillis() - startTime;
                    long elapsedSeconds = elapsedTimeMillis / 1000;
                    long minutes = elapsedSeconds / 60;
                    long seconds = elapsedSeconds % 60;
                    System.out.printf("\rElapsed Time: %02d:%02d - Waiting for
cluster...", minutes, seconds);
                    System.out.flush();

                    // Wait 1 second before the next status check
                    return CompletableFuture.runAsync(() -> {
                        try {
                            TimeUnit.SECONDS.sleep(1);
```

```
                        } catch (InterruptedException e) {
                            throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
                        }
                    }).thenCompose(ignored ->
checkClusterStatusAsync(clustersRequest, startTime));
                }
            }).exceptionally(exception -> {
                throw new RuntimeException("Failed to get cluster status: " +
exception.getMessage(), exception);
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeClusters를 참조하세요.

`DescribeStatement`

다음 코드 예시는 `DescribeStatement`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Checks the status of an SQL statement asynchronously and handles the
completion of the statement.
     *
     * @param sqlId the ID of the SQL statement to check
     * @return a {@link CompletableFuture} that completes when the SQL statement's
status is either "FINISHED" or "FAILED"
     */
    public CompletableFuture<Void> checkStatementAsync(String sqlId) {
        DescribeStatementRequest statementRequest =
DescribeStatementRequest.builder()
            .id(sqlId)
            .build();
```

```
        return getAsyncDataClient().describeStatement(statementRequest)
            .thenCompose(response -> {
                String status = response.statusAsString();
                logger.info("... Status: {} ", status);

                if ("FAILED".equals(status)) {
                    throw new RuntimeException("The Query Failed. Ending program");
                } else if ("FINISHED".equals(status)) {
                    return CompletableFuture.completedFuture(null);
                } else {
                    // Sleep for 1 second and recheck status
                    return CompletableFuture.runAsync(() -> {
                        try {
                            TimeUnit.SECONDS.sleep(1);
                        } catch (InterruptedException e) {
                            throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
                        }
                    }).thenCompose(ignore -> checkStatementAsync(sqlId)); //
Recursively call until status is FINISHED or FAILED
                }
            }).whenComplete((result, exception) -> {
                if (exception != null) {
                    // Handle exceptions
                    logger.info("Error: {} ", exception.getMessage());
                } else {
                    logger.info("The statement is finished!");
                }
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeStatement를 참조하세요.

`ExecuteStatement`

다음 코드 예시는 `ExecuteStatement`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

SQL 스테이트먼트를 실행하여 데이터베이스 테이블을 만듭니다.

```
   /**
    * Creates an asynchronous task to execute a SQL statement for creating a new
table.
    *
    * @param clusterId    the identifier of the Amazon Redshift cluster
    * @param databaseName the name of the database to create the table in
    * @param userName     the username to use for the database connection
    * @return a {@link CompletableFuture} that completes with the result of the SQL
statement execution
    * @throws RuntimeException if there is an error creating the table
    */
   public CompletableFuture<ExecuteStatementResponse> createTableAsync(String
clusterId, String databaseName, String userName) {
       ExecuteStatementRequest createTableRequest =
ExecuteStatementRequest.builder()
           .clusterIdentifier(clusterId)
           .dbUser(userName)
           .database(databaseName)
           .sql("CREATE TABLE Movies (" +
               "id INT PRIMARY KEY, " +
               "title VARCHAR(100), " +
               "year INT)")
           .build();

       return getAsyncDataClient().executeStatement(createTableRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   throw new RuntimeException("Error creating table: " +
exception.getMessage(), exception);
               } else {
                   logger.info("Table created: Movies");
               }
           });
```

```
    }
```

SQL 스테이트먼트를 실행하여 데이터베이스 테이블에 데이터를 삽입합니다.

```
    /**
     * Asynchronously pops a table from a JSON file.
     *
     * @param clusterId   the ID of the cluster
     * @param databaseName the name of the database
     * @param userName    the username
     * @param fileName    the name of the JSON file
     * @param number      the number of records to process
     * @return a CompletableFuture that completes with the number of records added
to the Movies table
     */
    public CompletableFuture<Integer> popTableAsync(String clusterId, String
databaseName, String userName, String fileName, int number) {
        return CompletableFuture.supplyAsync(() -> {
                try {
                    JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                    JsonNode rootNode = new ObjectMapper().readTree(parser);
                    Iterator<JsonNode> iter = rootNode.iterator();
                    return iter;
                } catch (IOException e) {
                    throw new RuntimeException("Failed to read or parse JSON file: "
+ e.getMessage(), e);
                }
            }).thenCompose(iter -> processNodesAsync(clusterId, databaseName,
userName, iter, number))
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    logger.info("Error {} ", exception.getMessage());
                } else {
                    logger.info("{} records were added to the Movies table." ,
result);
                }
            });
    }

    private CompletableFuture<Integer> processNodesAsync(String clusterId, String
databaseName, String userName, Iterator<JsonNode> iter, int number) {
```

```
        return CompletableFuture.supplyAsync(() -> {
            int t = 0;
            try {
                while (iter.hasNext()) {
                    if (t == number)
                        break;
                    JsonNode currentNode = 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);";
                    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);

                    ExecuteStatementRequest insertStatementRequest =
ExecuteStatementRequest.builder()
                        .clusterIdentifier(clusterId)
                        .sql(sqlStatement)
                        .database(databaseName)
                        .dbUser(userName)
                        .parameters(parameterList)
                        .build();

                    getAsyncDataClient().executeStatement(insertStatementRequest);
                    logger.info("Inserted: " + title + " (" + year + ")");
                    t++;
```

```
                }
            } catch (RedshiftDataException e) {
                throw new RuntimeException("Error inserting data: " +
e.getMessage(), e);
            }
            return t;
        });
    }
```

SQL 스테이트먼트를 실행하여 데이터베이스 테이블을 쿼리합니다.

```
    /**
     * Asynchronously queries movies by a given year from a Redshift database.
     *
     * @param database    the name of the database to query
     * @param dbUser      the user to connect to the database with
     * @param year        the year to filter the movies by
     * @param clusterId   the identifier of the Redshift cluster to connect to
     * @return a {@link CompletableFuture} containing the response ID of the
executed SQL statement
     */
    public CompletableFuture<String> queryMoviesByYearAsync(String database,
                                                            String dbUser,
                                                            int year,
                                                            String clusterId)
{

        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();

        return CompletableFuture.supplyAsync(() -> {
```

```
            try {
                ExecuteStatementResponse response =
getAsyncDataClient().executeStatement(statementRequest).join(); // Use join() to
wait for the result
                return response.id();
            } catch (RedshiftDataException e) {
                throw new RuntimeException("Error executing statement: " +
e.getMessage(), e);
            }
        }).exceptionally(exception -> {
            logger.info("Error: {}", exception.getMessage());
            return "";
        });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ExecuteStatement를 참조하세요.

`GetStatementResult`

다음 코드 예시는 `GetStatementResult`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

문 결과를 확인합니다.

```
    /**
     * Asynchronously retrieves the results of a statement execution.
     *
     * @param statementId the ID of the statement for which to retrieve the results
     * @return a {@link CompletableFuture} that completes when the statement result
has been processed
     */
    public CompletableFuture<Void> getResultsAsync(String statementId) {
        GetStatementResultRequest resultRequest =
GetStatementResultRequest.builder()
```

```
            .id(statementId)
            .build();

        return getAsyncDataClient().getStatementResult(resultRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    logger.info("Error getting statement result {} ",
exception.getMessage());
                    throw new RuntimeException("Error getting statement result: " +
exception.getMessage(), exception);
                }

                // Extract and print the field values using streams if the response
is valid.
                response.records().stream()
                    .flatMap(List::stream)
                    .map(Field::stringValue)
                    .filter(value -> value != null)
                    .forEach(value -> System.out.println("The Movie title field is "
+ value));

                return response;
            }).thenAccept(response -> {
                // Optionally add more logic here if needed after handling the
response
            });
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 GetStatementResult를 참조하세요.

`ListDatabases`

다음 코드 예시는 `ListDatabases`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Lists all databases asynchronously for the specified cluster, database user,
and database.
    * @param clusterId the identifier of the cluster to list databases for
    * @param dbUser the database user to use for the list databases request
    * @param database the database to list databases for
    * @return a {@link CompletableFuture} that completes when the database listing
is complete, or throws a {@link RuntimeException} if there was an error
    */
   public CompletableFuture<Void> listAllDatabasesAsync(String clusterId, String
dbUser, String database) {
       ListDatabasesRequest databasesRequest = ListDatabasesRequest.builder()
           .clusterIdentifier(clusterId)
           .dbUser(dbUser)
           .database(database)
           .build();

       // Asynchronous paginator for listing databases.
       ListDatabasesPublisher databasesPaginator =
getAsyncDataClient().listDatabasesPaginator(databasesRequest);
       CompletableFuture<Void> future = databasesPaginator.subscribe(response -> {
           response.databases().forEach(db -> {
               logger.info("The database name is {} ", db);
           });
       });

       // Return the future for asynchronous handling.
       return future.exceptionally(exception -> {
           throw new RuntimeException("Failed to list databases: " +
exception.getMessage(), exception);
       });
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListDatabases를 참조하시기 바랍니다.

`ModifyCluster`

다음 코드 예시는 `ModifyCluster`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

클러스터를 수정하세요.

```
   /**
    * Modifies an Amazon Redshift cluster asynchronously.
    *
    * @param clusterId the identifier of the cluster to be modified
    * @return a {@link CompletableFuture} that completes when the cluster
modification is complete
    */
   public CompletableFuture<ModifyClusterResponse> modifyClusterAsync(String
clusterId) {
       ModifyClusterRequest modifyClusterRequest = ModifyClusterRequest.builder()
           .clusterIdentifier(clusterId)
           .preferredMaintenanceWindow("wed:07:30-wed:08:00")
           .build();

       return getAsyncClient().modifyCluster(modifyClusterRequest)
           .whenComplete((clusterResponse, exception) -> {
               if (exception != null) {
                   if (exception.getCause() instanceof RedshiftException) {
                       logger.info("Error: {} ", exception.getMessage());
                   } else {
                       logger.info("Unexpected error: {} ",
exception.getMessage());
                   }
               } else {
                   logger.info("The modified cluster was successfully modified and
has "
                       + clusterResponse.cluster().preferredMaintenanceWindow() + "
as the maintenance window");
               }
           });
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ModifyCluster를 참조하세요.

시나리오

Amazon Redshift 데이터를 추적하는 웹 애플리케이션 만들기

다음 코드 예제에서는 Amazon Redshift 데이터베이스를 사용하는 작업 항목을 추적하고 보고하는 웹 애플리케이션을 만드는 방법을 보여줍니다.

SDK for Java 2.x

Amazon Redshift 데이터베이스에 저장된 작업 항목을 추적하고 보고하는 웹 애플리케이션을 만드는 방법을 보여줍니다.

Amazon Redshift 데이터를 쿼리하고 React 애플리케이션에서 사용하도록 Spring REST API를 설정하는 방법에 대한 지침과 전체 소스 코드는 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Redshift
- Amazon SES

Java 2.x용 SDK를 사용하는 Amazon Rekognition 예제

다음 코드 예제에서는 AWS SDK for Java 2.x Amazon Rekognition에서를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업
- 시나리오

작업

`CompareFaces`

다음 코드 예시는 CompareFaces의 사용 방법을 보여줍니다.

자세한 내용은 이미지에 있는 얼굴 비교를 참조하세요.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;
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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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: <bucketName> <sourceKey> <targetKey>

            Where:
                bucketName - The name of the S3 bucket where the images are stored.
```

```
                sourceKey  - The S3 key (file name) for the source image.
                targetKey  - The S3 key (file name) for the target image.
           """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceKey = args[1];
        String targetKey = args[2];

        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();
        compareTwoFaces(rekClient, bucketName, sourceKey, targetKey);
    }

   /**
    * Compares two faces from images stored in an Amazon S3 bucket using AWS
Rekognition.
    *
    * <p>This method takes two image keys from an S3 bucket and compares the faces
within them.
    * It prints out the confidence level of matched faces and reports the number of
unmatched faces.</p>
    *
    * @param rekClient   The {@link RekognitionClient} used to call AWS
Rekognition.
    * @param bucketName  The name of the S3 bucket containing the images.
    * @param sourceKey   The object key (file path) for the source image in the S3
bucket.
    * @param targetKey   The object key (file path) for the target image in the S3
bucket.
    * @throws RuntimeException If the Rekognition service returns an error.
    */
   public static void compareTwoFaces(RekognitionClient rekClient, String
bucketName, String sourceKey, String targetKey) {
        try {
            Float similarityThreshold = 70F;
            S3Object s3ObjectSource = S3Object.builder()
                    .bucket(bucketName)
```

```
                .name(sourceKey)
                .build();

        Image sourceImage = Image.builder()
                .s3Object(s3ObjectSource)
                .build();

        S3Object s3ObjectTarget = S3Object.builder()
                .bucket(bucketName)
                .name(targetKey)
                .build();

        Image targetImage = Image.builder()
                .s3Object(s3ObjectTarget)
                .build();

        CompareFacesRequest facesRequest = CompareFacesRequest.builder()
                .sourceImage(sourceImage)
                .targetImage(targetImage)
                .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> unmatchedFaces = compareFacesResult.unmatchedFaces();
        System.out.println("There were " + unmatchedFaces.size() + " face(s)
that did not match.");

       } catch (RekognitionException e) {
            System.err.println("Error comparing faces: " +
e.awsErrorDetails().errorMessage());
            throw new RuntimeException(e);
```

```
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CompareFaces를 참조하세요.

`CreateCollection`

다음 코드 예시는 `CreateCollection`의 사용 방법을 보여줍니다.

자세한 내용은 컬렉션 생성을 참조하세요.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Creating collection: " + collectionId);
        createMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    /**
     * Creates a new Amazon Rekognition collection.
     *
     * @param rekClient    the Amazon Rekognition client used to interact with the
Rekognition service
     * @param collectionId the unique identifier for the collection to be created
     */
    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);
```

```
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateCollection을 참조하세요.

`DeleteCollection`

다음 코드 예시는 `DeleteCollection`의 사용 방법을 보여줍니다.

자세한 내용은 컬렉션 삭제를 참조하세요.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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();
    }

    /**
     * Deletes an Amazon Rekognition collection.
     *
     * @param rekClient      An instance of the {@link RekognitionClient} class,
 which is used to interact with the Amazon Rekognition service.
     * @param collectionId   The ID of the collection to be deleted.
     */
    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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteCollection을 참조하세요.

`DeleteFaces`

다음 코드 예시는 `DeleteFaces`의 사용 방법을 보여줍니다.

자세한 내용은 컬렉션에서 얼굴 삭제를 참조하세요.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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 != 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("Deleting collection: " + collectionId);
        deleteFacesCollection(rekClient, collectionId, faceId);
        rekClient.close();
    }

    /**
     * Deletes a face from the specified Amazon Rekognition collection.
     *
     * @param rekClient     an instance of the Amazon Rekognition client
     * @param collectionId  the ID of the collection from which the face should be
deleted
     * @param faceId        the ID of the face to be deleted
     * @throws RekognitionException if an error occurs while deleting the face
     */
    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);
        }
    }
```

```
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteFaces를 참조하세요.

`DescribeCollection`

다음 코드 예시는 `DescribeCollection`의 사용 방법을 보여줍니다.

자세한 내용은 컬렉션 설명을 참조하세요.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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();
    }

    /**
     * Describes an Amazon Rekognition collection.
     *
     * @param rekClient         The Amazon Rekognition client used to make the
request.
     * @param collectionName    The name of the collection to describe.
     *
     * @throws RekognitionException If an error occurs while describing the
collection.
     */
    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);
        }
```

```
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeCollection을 참조하세요.

`DetectFaces`

다음 코드 예시는 `DetectFaces`의 사용 방법을 보여줍니다.

자세한 내용은 이미지에서 얼굴 감지를 참조하세요.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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:   <bucketName> <sourceImage>

            Where:
```

```
                bucketName = The name of the Amazon S3 bucket where the source image
is stored.
                sourceImage - The name of the source image file in the Amazon S3
bucket. (for example, pic1.png).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectFacesinImage(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Detects faces in an image stored in an Amazon S3 bucket using the Amazon
Rekognition service.
     *
     * @param rekClient    The Amazon Rekognition client used to interact with the
Rekognition service.
     * @param bucketName   The name of the Amazon S3 bucket where the source image
is stored.
     * @param sourceImage  The name of the source image file in the Amazon S3
bucket.
     */
    public static void detectFacesinImage(RekognitionClient rekClient, String
bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                .bucket(bucketName)
                .name(sourceImage)
                .build();

            Image targetImage = Image.builder()
                .s3Object(s3ObjectTarget)
                .build();
```

```
            DetectFacesRequest facesRequest = DetectFacesRequest.builder()
                .attributes(Attribute.ALL)
                .image(targetImage)
                .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 e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DetectFaces를 참조하세요.

`DetectLabels`

다음 코드 예시는 `DetectLabels`의 사용 방법을 보여줍니다.

자세한 내용은 이미지에서 레이블 감지를 참조하세요.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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.*;

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: <bucketName> <sourceImage>

            Where:
                bucketName - The name of the Amazon S3 bucket where the image is
 stored
                sourceImage - The name of the image file (for example, pic1.png).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0] ;
        String sourceImage = args[1] ;
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectImageLabels(rekClient, bucketName, sourceImage);
        rekClient.close();
```

```
    }

    /**
     * Detects the labels in an image stored in an Amazon S3 bucket using the Amazon
 Rekognition service.
     *
     * @param rekClient     the Amazon Rekognition client used to make the detection
 request
     * @param bucketName    the name of the Amazon S3 bucket where the image is
 stored
     * @param sourceImage   the name of the image file to be analyzed
     */
    public static void detectImageLabels(RekognitionClient rekClient, String
 bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

            Image souImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .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 e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DetectLabels를 참조하세요.

`DetectModerationLabels`

다음 코드 예시는 `DetectModerationLabels`의 사용 방법을 보여줍니다.

자세한 내용은 부적절한 이미지 감지를 참조하세요.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;

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:  <bucketName>  <sourceImage>

            Where:
```

```
                bucketName - The name of the S3 bucket where the images are stored.
                sourceImage - The name of the image (for example, pic1.png).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectModLabels(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Detects moderation labels in an image stored in an Amazon S3 bucket.
     *
     * @param rekClient      the Amazon Rekognition client to use for the detection
     * @param bucketName     the name of the Amazon S3 bucket where the image is
stored
     * @param sourceImage    the name of the image file to be analyzed
     *
     * @throws RekognitionException if there is an error during the image detection
process
     */
    public static void detectModLabels(RekognitionClient rekClient, String
bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

            Image targetImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();
```

```
            DetectModerationLabelsRequest moderationLabelsRequest =
 DetectModerationLabelsRequest.builder()
                    .image(targetImage)
                    .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 e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DetectModerationLabels를 참조하세요.

`DetectText`

다음 코드 예시는 `DetectText`의 사용 방법을 보여줍니다.

자세한 내용은 이미지에서 텍스트 감지를 참조하세요.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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.*;

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 = "\n" +
            "Usage:   <bucketName> <sourceImage>\n" +
            "\n" +
            "Where:\n" +
            "   bucketName - The name of the S3 bucket where the image is stored\n"
 +
            "   sourceImage - The path to the image that contains text (for example,
 pic1.png). \n";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectTextLabels(rekClient, bucketName, sourceImage);
        rekClient.close();
```

```
    }

    /**
     * Detects text labels in an image stored in an S3 bucket using Amazon
Rekognition.
     *
     * @param rekClient    an instance of the Amazon Rekognition client
     * @param bucketName   the name of the S3 bucket where the image is stored
     * @param sourceImage  the name of the image file in the S3 bucket
     * @throws RekognitionException if an error occurs while calling the Amazon
Rekognition API
     */
    public static void detectTextLabels(RekognitionClient rekClient, String
bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

            Image souImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .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 e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
```

```
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DetectText를 참조하세요.

`IndexFaces`

다음 코드 예시는 `IndexFaces`의 사용 방법을 보여줍니다.

자세한 내용은 컬렉션에 얼굴 추가를 참조하세요.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;
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> <bucketName>

            Where:
                collectionName - The name of the collection.
                sourceImage - The name of the image (for example, pic1.png).
                bucketName - The name of the S3 bucket.
```

```
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String sourceImage = args[1];
        String bucketName = args[2];;
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        addToCollection(rekClient, collectionId, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Adds a face from an image to an Amazon Rekognition collection.
     *
     * @param rekClient     the Amazon Rekognition client
     * @param collectionId  the ID of the collection to add the face to
     * @param bucketName    the name of the Amazon S3 bucket containing the image
     * @param sourceImage   the name of the image file to add to the collection
     * @throws RekognitionException if there is an error while interacting with the
Amazon Rekognition service
     */
    public static void addToCollection(RekognitionClient rekClient, String
collectionId, String bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

            Image targetImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();

            IndexFacesRequest facesRequest = IndexFacesRequest.builder()
                    .collectionId(collectionId)
                    .image(targetImage)
```

```
                    .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 e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 IndexFaces를 참조하세요.

`ListCollections`

다음 코드 예시는 `ListCollections`의 사용 방법을 보여줍니다.

자세한 내용은 컬렉션 나열을 참조하세요.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListCollections를 참조하세요.

`ListFaces`

다음 코드 예시는 `ListFaces`의 사용 방법을 보여줍니다.

자세한 내용은 컬렉션 내 얼굴 나열을 참조하세요.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListFaces를 참조하세요.

`RecognizeCelebrities`

다음 코드 예시는 `RecognizeCelebrities`의 사용 방법을 보여줍니다.

자세한 내용은 이미지에서 유명인 인식을 참조하세요.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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.*;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * 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:   <bucketName> <sourceImage>

                Where:
                   bucketName - The name of the S3 bucket where the images are
 stored.
                   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 bucketName = args[0];;
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Locating celebrities in " + sourceImage);
        recognizeAllCelebrities(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Recognizes all celebrities in an image stored in an Amazon S3 bucket.
     *
     * @param rekClient    the Amazon Rekognition client used to perform the
 celebrity recognition operation
     * @param bucketName   the name of the Amazon S3 bucket where the source image
 is stored
     * @param sourceImage  the name of the source image file stored in the Amazon S3
 bucket
     */
    public static void recognizeAllCelebrities(RekognitionClient rekClient, String
 bucketName, String sourceImage) {
```

```
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                .bucket(bucketName)
                .name(sourceImage)
                .build();

            Image souImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .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 e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 RecognizeCelebrities를 참조하세요.

SearchFaces

다음 코드 예시는 SearchFaces의 사용 방법을 보여줍니다.

자세한 내용은 얼굴 검색(얼굴 ID)을 참조하세요.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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_WEST_2;
        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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 SearchFaces를 참조하세요.

`SearchFacesByImage`

다음 코드 예시는 `SearchFacesByImage`의 사용 방법을 보여줍니다.

자세한 내용은 얼굴 검색(이미지)을 참조하세요.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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_WEST_2;
        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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 SearchFacesByImage를 참조하세요.

시나리오

사진을 관리하기 위한 서버리스 애플리케이션 만들기

다음 코드 예시에서는 사용자가 레이블을 사용하여 사진을 관리할 수 있는 서버리스 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 AWS 커뮤니티의 게시물을 참조하세요.

이 예제에서 사용되는 서비스

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

- Amazon SNS

이미지에서 PPE 감지

다음 코드 예제에서는 Amazon Rekognition을 사용하여 이미지에서 개인 보호 장비(PPE)를 감지하는 앱을 구축하는 방법을 보여줍니다.

SDK for Java 2.x

개인 보호 장비로 이미지를 감지하는 AWS Lambda 함수를 생성하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

동영상 내 정보 감지

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- Amazon Rekognition 작업을 시작하여 동영상에서 사람, 사물, 텍스트와 같은 요소를 탐지하세요.
- 작업이 완료될 때까지 작업 상태를 확인하세요.
- 각 작업에서 감지한 요소의 목록을 출력합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon S3 버킷에 있는 동영상에서 유명인사의 결과를 가져옵니다.

```
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);
        }
    }
}
```

레이블 감지 작업을 통해 동영상의 레이블을 감지합니다.

```
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);
        }
    }
}
```

Amazon S3 버킷에 저장된 동영상에서 얼굴을 감지합니다.

```
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);
        }
    }
}
```

Amazon S3 버킷에 저장된 동영상에서 부적절하거나 불쾌감을 주는 콘텐츠를 감지합니다.

```
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);
        }
    }
}
```

Amazon S3 버킷에 저장된 동영상에서 기술적 큐 세그먼트와 샷 감지 세그먼트를 감지합니다.

```
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);
        }
    }
}
```

Amazon S3 버킷에 저장된 동영상에서 텍스트를 감지합니다.

```
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);
        }
    }
}
```

Amazon S3 버킷에 저장된 동영상에서 사람을 감지합니다.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - GetCelebrityRecognition
  - GetContentModeration
  - GetLabelDetection
  - GetPersonTracking
  - GetSegmentDetection
  - GetTextDetection
  - StartCelebrityRecognition
  - StartContentModeration
  - StartLabelDetection
  - StartPersonTracking
  - StartSegmentDetection
  - StartTextDetection

이미지에서 객체 감지

다음 코드 예제에서는 Amazon Rekognition을 사용하여 이미지에서 범주별 객체를 감지하는 앱을 구축하는 방법을 보여줍니다.

SDK for Java 2.x

Amazon Rekognition을 사용하여 Amazon Simple Storage Service(Amazon S3) 버킷에 있는 이미지에서 범주별로 객체를 식별하기 위해 Amazon Rekognition을 사용하여 앱을 만드는 방법을 보여줍니다. 이 앱은 Amazon Simple Email Service(Amazon SES)를 사용하여 결과와 함께 이메일 알림을 관리자에게 보냅니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Rekognition
- Amazon S3
- Amazon SES

동영상에서 사람과 객체 감지

다음 코드 예제에서는 Amazon Rekognition을 사용하여 동영상에서 사람과 객체를 감지하는 방법을 보여줍니다.

SDK for Java 2.x

Amazon Rekognition Java API를 사용하여 Amazon Simple Storage Service(Amazon S3) 버킷에 있는 동영상에서 얼굴과 객체를 감지하기 위한 앱을 만드는 방법을 보여줍니다. 이 앱은 Amazon Simple Email Service(Amazon SES)를 사용하여 결과와 함께 이메일 알림을 관리자에게 보냅니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Java 2.x용 SDK를 사용하는 Route 53 도메인 등록 예제

다음 코드 예제에서는 Route 53 도메인 등록과 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시작하기

Route 53 도메인 등록 소개

다음 코드 예제에서는 Route 53 도메인 등록 사용을 시작하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListPrices를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 현재 도메인과 작년의 작업을 나열합니다.
- 작년의 결제 내역과 도메인 유형의 가격을 봅니다.

- 도메인 제안을 가져옵니다.
- 도메인 가용성 및 이전 가능성을 확인합니다.
- 선택 사항으로 도메인 등록을 요청할 수도 있습니다.
- 작업 세부 정보를 가져옵니다.
- 선택 사항으로 도메인 세부 정보를 가져올 수 있습니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including 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);
       }
   }
```

```
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - CheckDomainAvailability
  - CheckDomainTransferability
  - GetDomainDetail
  - GetDomainSuggestions
  - GetOperationDetail
  - ListDomains
  - ListOperations
  - ListPrices
  - RegisterDomain
  - ViewBilling

작업

`CheckDomainAvailability`

다음 코드 예시는 `CheckDomainAvailability`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조에서 CheckDomainAvailability를 참조하세요.

`CheckDomainTransferability`

다음 코드 예시는 `CheckDomainTransferability`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CheckDomainTransferability를 참조하세요.

`GetDomainDetail`

다음 코드 예시는 `GetDomainDetail`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetDomainDetail을 참조하세요.

`GetDomainSuggestions`

다음 코드 예시는 `GetDomainSuggestions`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetDomainSuggestions를 참조하세요.

GetOperationDetail

다음 코드 예시는 GetOperationDetail의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetOperationDetail을 참조하세요.

ListDomains

다음 코드 예시는 ListDomains의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListDomains를 참조하세요.

`ListOperations`

다음 코드 예시는 `ListOperations`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListOperations를 참조하세요.

`ListPrices`

다음 코드 예시는 `ListPrices`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListPrices를 참조하세요.

`RegisterDomain`

다음 코드 예시는 `RegisterDomain`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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 "";
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 RegisterDomain을 참조하세요.

`ViewBilling`

다음 코드 예시는 `ViewBilling`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ViewBilling을 참조하세요.

Java 2.x용 SDK를 사용하는 Amazon S3 예제

다음 코드 예제에서는 AWS SDK for Java 2.x Amazon S3에서를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시작하기
- 기본 사항
- 작업
- 시나리오
- 서버리스 예제

시작하기

Hello Amazon S3

다음 코드 예제에서는 Amazon S3 사용을 시작하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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);
    }

    /**
     * Lists all the S3 buckets associated with the provided AWS S3 client.
     *
     * @param s3 the S3Client instance used to interact with the AWS S3 service
     */
    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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListBuckets를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 버킷을 만들고 버킷에 파일을 업로드합니다.
- 버킷에서 객체를 다운로드합니다.
- 버킷의 하위 폴더에 객체를 복사합니다.
- 버킷의 객체를 나열합니다.
- 버킷 객체와 버킷을 삭제합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

시나리오 예제.

```
import java.io.IOException;
```

```
import java.util.Scanner;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
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
 *
 * 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. Copy the object to another Amazon S3 bucket using multi copy.
 * 8. Deletes the object from the Amazon S3 bucket.
 * 9. Deletes the Amazon S3 bucket.
 */

public class S3Scenario {

    public static Scanner scanner = new Scanner(System.in);
    static S3Actions s3Actions = new S3Actions();
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final Logger logger = LoggerFactory.getLogger(S3Scenario.class);
    public static void main(String[] args) throws IOException {
        final String usage = """
            Usage:
               <bucketName> <key> <objectPath> <savePath> <toBucket>

            Where:
                bucketName - The name of the  S3 bucket.
                key - The unique identifier for the object stored in the S3 bucket.
```

```
                objectPath - The full file path of the object within the S3 bucket
(e.g., "documents/reports/annual_report.pdf").
                savePath - The local file path where the object will be downloaded
and saved (e.g., "C:/Users/username/Downloads/annual_report.pdf").
                toBucket - The name of the S3 bucket to which the object will be
copied.
            """;

        if (args.length != 5) {
            logger.info(usage);
            return;
        }

        String bucketName = args[0];
        String key = args[1];
        String objectPath = args[2];
        String savePath = args[3];
        String toBucket = args[4];

        logger.info(DASHES);
        logger.info("Welcome to the Amazon Simple Storage Service (S3) example
scenario.");
        logger.info("""
            Amazon S3 is a highly scalable and durable object storage
            service provided by Amazon Web Services (AWS). It is designed to store
and retrieve
            any amount of data, from anywhere on the web, at any time.

            The `S3AsyncClient` interface in the AWS SDK for Java 2.x provides a set
of methods to
            programmatically interact with the Amazon S3 (Simple Storage Service)
service. This allows
            developers to automate the management and manipulation of S3 buckets and
objects as
            part of their application deployment pipelines. With S3, teams can focus
on building
            and deploying their applications without having to worry about the
underlying storage
            infrastructure required to host and manage large amounts of data.

            This scenario walks you through how to perform key operations for this
service.
            Let's get started...
            """);
```

```
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        try {
            // Run the methods that belong to this scenario.
            runScenario(bucketName, key, objectPath, savePath, toBucket);

        } catch (Throwable rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }

    private static void runScenario(String bucketName, String key, String
objectPath, String savePath, String toBucket) throws Throwable {
        logger.info(DASHES);
        logger.info("1. Create an Amazon S3 bucket.");
        try {
            CompletableFuture<Void> future =
s3Actions.createBucketAsync(bucketName);
            future.join();
            waitForInputToContinue(scanner);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;

        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Upload a local file to the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
```

```
        try {
            CompletableFuture<PutObjectResponse> future =
s3Actions.uploadLocalFileAsync(bucketName, key, objectPath);
            future.join();
            logger.info("File uploaded successfully to {}/{}", bucketName, key);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);


        logger.info(DASHES);
        logger.info("3. Download the object to another local file.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.getObjectBytesAsync(bucketName, key, savePath);
            future.join();
            logger.info("Successfully obtained bytes from S3 object and wrote to
file {}", savePath);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
```

```
        logger.info("4. Perform a multipart upload.");
        waitForInputToContinue(scanner);
        String multipartKey = "multiPartKey";
        try {
            // Call the multipartUpload method
            CompletableFuture<Void> future = s3Actions.multipartUpload(bucketName,
multipartKey);
            future.join();
            logger.info("Multipart upload completed successfully for bucket '{}' and
key '{}'", bucketName, multipartKey);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. List all objects located in the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.listAllObjectsAsync(bucketName);
            future.join();
            logger.info("Object listing completed successfully.");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
```

```
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. Copy the object to another Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
s3Actions.copyBucketObjectAsync(bucketName, key, toBucket);
            String result = future.join();
            logger.info("Copy operation result: {}", result);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Copy the object to another Amazon S3 bucket using multi
copy.");
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<String> future = s3Actions.performMultiCopy(toBucket,
bucketName, key);
            String result = future.join();
            logger.info("Copy operation result: {}", result);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
```

```
        waitForInputToContinue(scanner);
        logger.info(DASHES);


        logger.info(DASHES);
        logger.info("8. Delete objects from the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.deleteObjectFromBucketAsync(bucketName, key);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        try {
            CompletableFuture<Void> future =
s3Actions.deleteObjectFromBucketAsync(bucketName, "multiPartKey");
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("9. Delete the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
```

```
            CompletableFuture<Void> future =
 s3Actions.deleteBucketAsync(bucketName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
 s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("You successfully completed the Amazon S3 scenario.");
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

작업이 포함된 래퍼 클래스.

```
public class S3Actions {

    private static final Logger logger = LoggerFactory.getLogger(S3Actions.class);
    private static S3AsyncClient s3AsyncClient;

    public static S3AsyncClient getAsyncClient() {
        if (s3AsyncClient == null) {
            /*
            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.
                .retryStrategy(RetryMode.STANDARD)
                .build();

            s3AsyncClient = S3AsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return s3AsyncClient;
    }
```

```
   /**
    * Creates an S3 bucket asynchronously.
    *
    * @param bucketName the name of the S3 bucket to create
    * @return a {@link CompletableFuture} that completes when the bucket is created
and ready
    * @throws RuntimeException if there is a failure while creating the bucket
    */
   public CompletableFuture<Void> createBucketAsync(String bucketName) {
       CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
           .bucket(bucketName)
           .build();

       CompletableFuture<CreateBucketResponse> response =
getAsyncClient().createBucket(bucketRequest);
       return response.thenCompose(resp -> {
           S3AsyncWaiter s3Waiter = getAsyncClient().waiter();
           HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
               .bucket(bucketName)
               .build();

           CompletableFuture<WaiterResponse<HeadBucketResponse>>
waiterResponseFuture =
               s3Waiter.waitUntilBucketExists(bucketRequestWait);
           return waiterResponseFuture.thenAccept(waiterResponse -> {
               waiterResponse.matched().response().ifPresent(headBucketResponse ->
{
                   logger.info(bucketName + " is ready");
               });
           });
       }).whenComplete((resp, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to create bucket", ex);
           }
       });
   }


   /**
    * Uploads a local file to an AWS S3 bucket asynchronously.
    *
    * @param bucketName the name of the S3 bucket to upload the file to
```

```
    * @param key        the key (object name) to use for the uploaded file
    * @param objectPath the local file path of the file to be uploaded
    * @return a {@link CompletableFuture} that completes with the {@link
PutObjectResponse} when the upload is successful, or throws a {@link
RuntimeException} if the upload fails
    */
   public CompletableFuture<PutObjectResponse> uploadLocalFileAsync(String
bucketName, String key, String objectPath) {
       PutObjectRequest objectRequest = PutObjectRequest.builder()
           .bucket(bucketName)
           .key(key)
           .build();

       CompletableFuture<PutObjectResponse> response =
getAsyncClient().putObject(objectRequest,
AsyncRequestBody.fromFile(Paths.get(objectPath)));
       return response.whenComplete((resp, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to upload file", ex);
           }
       });
   }


   /**
    * Asynchronously retrieves the bytes of an object from an Amazon S3 bucket and
writes them to a local file.
    *
    * @param bucketName the name of the S3 bucket containing the object
    * @param keyName    the key (or name) of the S3 object to retrieve
    * @param path       the local file path where the object's bytes will be
written
    * @return a {@link CompletableFuture} that completes when the object bytes have
been written to the local file
    */
   public CompletableFuture<Void> getObjectBytesAsync(String bucketName, String
keyName, String path) {
       GetObjectRequest objectRequest = GetObjectRequest.builder()
           .key(keyName)
           .bucket(bucketName)
           .build();

       CompletableFuture<ResponseBytes<GetObjectResponse>> response =
getAsyncClient().getObject(objectRequest, AsyncResponseTransformer.toBytes());
```

```
        return response.thenAccept(objectBytes -> {
            try {
                byte[] data = objectBytes.asByteArray();
                Path filePath = Paths.get(path);
                Files.write(filePath, data);
                logger.info("Successfully obtained bytes from an S3 object");
            } catch (IOException ex) {
                throw new RuntimeException("Failed to write data to file", ex);
            }
        }).whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to get object bytes from S3",
ex);
            }
        });
    }


    /**
     * Asynchronously lists all objects in the specified S3 bucket.
     *
     * @param bucketName the name of the S3 bucket to list objects for
     * @return a {@link CompletableFuture} that completes when all objects have been
listed
     */
    public CompletableFuture<Void> listAllObjectsAsync(String bucketName) {
        ListObjectsV2Request initialRequest = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .maxKeys(1)
            .build();

        ListObjectsV2Publisher paginator =
getAsyncClient().listObjectsV2Paginator(initialRequest);
        return paginator.subscribe(response -> {
            response.contents().forEach(s3Object -> {
                logger.info("Object key: " + s3Object.key());
            });
        }).thenRun(() -> {
            logger.info("Successfully listed all objects in the bucket: " +
bucketName);
        }).exceptionally(ex -> {
            throw new RuntimeException("Failed to list objects", ex);
        });
    }
```

```
   /**
    * Asynchronously copies an object from one S3 bucket to another.
    *
    * @param fromBucket the name of the source S3 bucket
    * @param objectKey  the key (name) of the object to be copied
    * @param toBucket   the name of the destination S3 bucket
    * @return a {@link CompletableFuture} that completes with the copy result as a
{@link String}
    * @throws RuntimeException if the URL could not be encoded or an S3 exception
occurred during the copy
    */
   public CompletableFuture<String> copyBucketObjectAsync(String fromBucket, String
objectKey, String toBucket) {
       CopyObjectRequest copyReq = CopyObjectRequest.builder()
           .sourceBucket(fromBucket)
           .sourceKey(objectKey)
           .destinationBucket(toBucket)
           .destinationKey(objectKey)
           .build();

       CompletableFuture<CopyObjectResponse> response =
getAsyncClient().copyObject(copyReq);
       response.whenComplete((copyRes, ex) -> {
           if (copyRes != null) {
               logger.info("The " + objectKey + " was copied to " + toBucket);
           } else {
               throw new RuntimeException("An S3 exception occurred during copy",
ex);
           }
       });

       return response.thenApply(CopyObjectResponse::copyObjectResult)
           .thenApply(Object::toString);
   }

   /**
    * Performs a multipart upload to an Amazon S3 bucket.
    *
    * @param bucketName the name of the S3 bucket to upload the file to
    * @param key        the key (name) of the file to be uploaded
    * @return a {@link CompletableFuture} that completes when the multipart upload
is successful
```

```
     */
    public CompletableFuture<Void> multipartUpload(String bucketName, String key) {
        int mB = 1024 * 1024;

        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        return getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
            .thenCompose(createResponse -> {
                String uploadId = createResponse.uploadId();
                System.out.println("Upload ID: " + uploadId);

                // Upload part 1.
                UploadPartRequest uploadPartRequest1 = UploadPartRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .partNumber(1)
                    .contentLength((long) (5 * mB)) // Specify the content length
                    .build();

                CompletableFuture<CompletedPart> part1Future =
getAsyncClient().uploadPart(uploadPartRequest1,
                        AsyncRequestBody.fromByteBuffer(getRandomByteBuffer(5 *
mB)))
                    .thenApply(uploadPartResponse -> CompletedPart.builder()
                        .partNumber(1)
                        .eTag(uploadPartResponse.eTag())
                        .build());

                // Upload part 2.
                UploadPartRequest uploadPartRequest2 = UploadPartRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .partNumber(2)
                    .contentLength((long) (3 * mB))
                    .build();

                CompletableFuture<CompletedPart> part2Future =
getAsyncClient().uploadPart(uploadPartRequest2,
```

```
                        AsyncRequestBody.fromByteBuffer(getRandomByteBuffer(3 *
mB)))
                    .thenApply(uploadPartResponse -> CompletedPart.builder()
                        .partNumber(2)
                        .eTag(uploadPartResponse.eTag())
                        .build());

                // Combine the results of both parts.
                return CompletableFuture.allOf(part1Future, part2Future)
                    .thenCompose(v -> {
                        CompletedPart part1 = part1Future.join();
                        CompletedPart part2 = part2Future.join();

                        CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
                            .parts(part1, part2)
                            .build();

                        CompleteMultipartUploadRequest
completeMultipartUploadRequest = CompleteMultipartUploadRequest.builder()
                            .bucket(bucketName)
                            .key(key)
                            .uploadId(uploadId)
                            .multipartUpload(completedMultipartUpload)
                            .build();

                        // Complete the multipart upload
                        return
getAsyncClient().completeMultipartUpload(completeMultipartUploadRequest);
                    });
            })
            .thenAccept(response -> System.out.println("Multipart upload completed
successfully"))
            .exceptionally(ex -> {
                System.err.println("Failed to complete multipart upload: " +
ex.getMessage());
                throw new RuntimeException(ex);
            });
    }


    /**
     * Deletes an object from an S3 bucket asynchronously.
     *
```

```
    * @param bucketName the name of the S3 bucket
    * @param key        the key (file name) of the object to be deleted
    * @return a {@link CompletableFuture} that completes when the object has been
deleted
    */
   public CompletableFuture<Void> deleteObjectFromBucketAsync(String bucketName,
String key) {
       DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
           .bucket(bucketName)
           .key(key)
           .build();

       CompletableFuture<DeleteObjectResponse> response =
getAsyncClient().deleteObject(deleteObjectRequest);
       response.whenComplete((deleteRes, ex) -> {
           if (deleteRes != null) {
               logger.info(key + " was deleted");
           } else {
               throw new RuntimeException("An S3 exception occurred during delete",
ex);
           }
       });

       return response.thenApply(r -> null);
   }


   /**
    * Deletes an S3 bucket asynchronously.
    *
    * @param bucket the name of the bucket to be deleted
    * @return a {@link CompletableFuture} that completes when the bucket deletion
is successful, or throws a {@link RuntimeException}
    * if an error occurs during the deletion process
    */
   public CompletableFuture<Void> deleteBucketAsync(String bucket) {
       DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
           .bucket(bucket)
           .build();

       CompletableFuture<DeleteBucketResponse> response =
getAsyncClient().deleteBucket(deleteBucketRequest);
       response.whenComplete((deleteRes, ex) -> {
           if (deleteRes != null) {
```

```
                logger.info(bucket + " was deleted.");
            } else {
                throw new RuntimeException("An S3 exception occurred during bucket
deletion", ex);
            }
        });
        return response.thenApply(r -> null);
    }

    public CompletableFuture<String> performMultiCopy(String toBucket, String
bucketName, String key) {
        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
            .bucket(toBucket)
            .key(key)
            .build();

        getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
            .thenApply(createMultipartUploadResponse -> {
                String uploadId = createMultipartUploadResponse.uploadId();
                System.out.println("Upload ID: " + uploadId);

                UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
                    .sourceBucket(bucketName)
                    .destinationBucket(toBucket)
                    .sourceKey(key)
                    .destinationKey(key)
                    .uploadId(uploadId)  // Use the valid uploadId.
                    .partNumber(1)  // Ensure the part number is correct.
                    .copySourceRange("bytes=0-1023")  // Adjust range as needed
                    .build();

                return getAsyncClient().uploadPartCopy(uploadPartCopyRequest);
            })
            .thenCompose(uploadPartCopyFuture -> uploadPartCopyFuture)
            .whenComplete((uploadPartCopyResponse, exception) -> {
                if (exception != null) {
                    // Handle any exceptions.
                    logger.error("Error during upload part copy: " +
exception.getMessage());
                } else {
                    // Successfully completed the upload part copy.
```

```
                    System.out.println("Upload Part Copy completed successfully.
 ETag: " + uploadPartCopyResponse.copyPartResult().eTag());
                }
            });
        return null;
    }

    private static ByteBuffer getRandomByteBuffer(int size) {
        ByteBuffer buffer = ByteBuffer.allocate(size);
        for (int i = 0; i < size; i++) {
            buffer.put((byte) (Math.random() * 256));
        }
        buffer.flip();
        return buffer;
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 항목을 참조하세요.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

작업

`AbortMultipartUpload`

다음 코드 예시는 `AbortMultipartUpload`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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.AbortMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadResponse;
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.LifecycleRule;
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.PutBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
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.services.sts.StsClient;
import software.amazon.awssdk.utils.builder.SdkBuilder;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.time.Duration;
import java.time.Instant;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;
```

```
import java.util.Objects;
import java.util.UUID;

import static software.amazon.awssdk.transfer.s3.SizeConstant.KB;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class AbortMultipartUploadExamples {
    static final String bucketName = "amzn-s3-demo-bucket" + UUID.randomUUID(); //
 Change bucket name.
    static final String key = UUID.randomUUID().toString();
    static final String classPathFilePath = "/multipartUploadFiles/s3-
userguide.pdf";
    static final String filePath = getFullFilePath(classPathFilePath);
    static final S3Client s3Client = S3Client.create();
    private static final Logger logger =
 LoggerFactory.getLogger(AbortMultipartUploadExamples.class);
    private static String accountId = getAccountId();

    public static void main(String[] args) {
        doAbortIncompleteMultipartUploadsFromList();
        doAbortMultipartUploadUsingUploadId();
        doAbortIncompleteMultipartUploadsOlderThan();
        doAbortMultipartUploadsUsingLifecycleConfig();
    }

    // A wrapper method that sets up the multipart upload environment for
 abortIncompleteMultipartUploadsFromList().
    public static void doAbortIncompleteMultipartUploadsFromList() {
        createBucket();
        initiateAndInterruptMultiPartUpload("uploadThread");
        abortIncompleteMultipartUploadsFromList();
        deleteResources();
    }

    /**
     * Aborts all incomplete multipart uploads from the specified S3 bucket.
```

```
    * <p>
    * This method retrieves a list of all incomplete multipart uploads in the
specified S3 bucket,
    * and then aborts each of those uploads.
    */
   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);
           }
       }
   }

   // A wrapper method that sets up the multipart upload environment for
abortIncompleteMultipartUploadsOlderThan().
   static void doAbortIncompleteMultipartUploadsOlderThan() {
       createBucket();
       Instant secondUploadInstant = initiateAndInterruptTwoUploads();
       abortIncompleteMultipartUploadsOlderThan(secondUploadInstant);
       deleteResources();
   }

   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);
                }
            }
        }
    }

    // A wrapper method that sets up the multipart upload environment for
abortMultipartUploadUsingUploadId().
    static void doAbortMultipartUploadUsingUploadId() {
        createBucket();
        try {
            abortMultipartUploadUsingUploadId();
        } catch (S3Exception e) {
            logger.error(e.getMessage());
        } finally {
            deleteResources();
        }
    }
```

```
    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);
        }
    }

    // A wrapper method that sets up the multipart upload environment for
abortMultipartUploadsUsingLifecycleConfig().
    static void doAbortMultipartUploadsUsingLifecycleConfig() {
        createBucket();
        try {
            abortMultipartUploadsUsingLifecycleConfig();
        } catch (S3Exception e) {
            logger.error(e.getMessage());
        } finally {
            deleteResources();
        }
    }

    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());
        }
    }

    /************************
     Multipart upload methods
     ************************/

    static void initiateAndInterruptMultiPartUpload(String threadName) {
        Runnable upload = () -> {
            try {
                AbortMultipartUploadExamples.doMultipartUpload();
            } catch (SdkException e) {
                logger.error(e.getMessage());
            }
        };
        Thread uploadThread = new Thread(upload, threadName);
        uploadThread.start();
        try {
            Thread.sleep(Duration.ofSeconds(1).toMillis()); // Give the multipart
upload time to register.
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
        }
        uploadThread.interrupt();
    }

    static Instant initiateAndInterruptTwoUploads() {
        Instant firstUploadInstant = Instant.now();
        initiateAndInterruptMultiPartUpload("uploadThread1");
        try {
            Thread.sleep(Duration.ofSeconds(5).toMillis());
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
        }
        Instant secondUploadInstant = Instant.now();
        initiateAndInterruptMultiPartUpload("uploadThread2");
        return secondUploadInstant;
    }

    static void doMultipartUpload() {
        String uploadId = step1CreateMultipartUpload();
```

```
        List<CompletedPart> completedParts = step2UploadParts(uploadId);
        step3CompleteMultipartUpload(uploadId, completedParts);
    }

    static String step1CreateMultipartUpload() {
        CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
            .bucket(bucketName)
            .key(key));
        return createMultipartUploadResponse.uploadId();
    }

    static List<CompletedPart> step2UploadParts(String uploadId) {
        int partNumber = 1;
        List<CompletedPart> completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(Long.valueOf(1024 * KB).intValue());

        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);
                logger.info("Part {} upload", partNumber);
```

```
                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (IOException | S3Exception e) {
            logger.error(e.getMessage());
            return null;
        }
        return completedParts;
    }

    static void step3CompleteMultipartUpload(String uploadId, List<CompletedPart>
completedParts) {
        s3Client.completeMultipartUpload(b -> b
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
    }

    static String startUploadReturningUploadId() {
        String uploadId = step1CreateMultipartUpload();
        doMultipartUploadWithUploadId(uploadId);
        return uploadId;

    }

    static void doMultipartUploadWithUploadId(String uploadId) {
        new Thread(() -> {
            try {
                List<CompletedPart> completedParts = step2UploadParts(uploadId);
                step3CompleteMultipartUpload(uploadId, completedParts);
            } catch (SdkException e) {
                logger.error(e.getMessage());
            }
        }, "upload thread").start();
        try {
            Thread.sleep(Duration.ofSeconds(2L).toMillis());
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
            System.exit(1);
        }
    }
```

```
    /************************
     Resource handling methods
     ************************/

    static void createBucket() {
        logger.info("Creating bucket: [{}]", bucketName);
        s3Client.createBucket(b -> b.bucket(bucketName));
        try (S3Waiter s3Waiter = s3Client.waiter()) {
            s3Waiter.waitUntilBucketExists(b -> b.bucket(bucketName));
        }
        logger.info("Bucket created.");
    }

    static void deleteResources() {
        logger.info("Deleting resources ...");
        s3Client.deleteObject(b -> b.bucket(bucketName).key(key));
        s3Client.deleteBucket(b -> b.bucket(bucketName));
        try (S3Waiter s3Waiter = s3Client.waiter()) {
            s3Waiter.waitUntilBucketNotExists(b -> b.bucket(bucketName));
        }
        logger.info("Resources deleted.");
    }

    private static String getAccountId() {
        try (StsClient stsClient = StsClient.create()) {
            return stsClient.getCallerIdentity().account();
        }
    }

    static String getFullFilePath(String filePath) {
        URL uploadDirectoryURL = PerformMultiPartUpload.class.getResource(filePath);
        String fullFilePath;
        try {
            fullFilePath =
 Objects.requireNonNull(uploadDirectoryURL).toURI().getPath();
        } catch (URISyntaxException e) {
            throw new RuntimeException(e);
        }
        return fullFilePath;
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 AbortMultipartUpload를 참조하세요.

`CopyObject`

다음 코드 예시는 `CopyObject`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

S3Client를 사용하여 객체를 복사합니다.

```
   /**
    * Asynchronously copies an object from one S3 bucket to another.
    *
    * @param fromBucket the name of the source S3 bucket
    * @param objectKey  the key (name) of the object to be copied
    * @param toBucket   the name of the destination S3 bucket
    * @return a {@link CompletableFuture} that completes with the copy result as a
{@link String}
    * @throws RuntimeException if the URL could not be encoded or an S3 exception
occurred during the copy
    */
   public CompletableFuture<String> copyBucketObjectAsync(String fromBucket, String
objectKey, String toBucket) {
       CopyObjectRequest copyReq = CopyObjectRequest.builder()
           .sourceBucket(fromBucket)
           .sourceKey(objectKey)
           .destinationBucket(toBucket)
           .destinationKey(objectKey)
           .build();

       CompletableFuture<CopyObjectResponse> response =
getAsyncClient().copyObject(copyReq);
       response.whenComplete((copyRes, ex) -> {
           if (copyRes != null) {
               logger.info("The " + objectKey + " was copied to " + toBucket);
```

```
            } else {
                throw new RuntimeException("An S3 exception occurred during copy",
 ex);
            }
        });

        return response.thenApply(CopyObjectResponse::copyObjectResult)
            .thenApply(Object::toString);
    }
```

S3TransferManager를 사용하여 한 버킷에서 다른 버킷으로 객체를 복사합니다. 파일 전체를 보고 테스트합니다.

```
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();
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CopyObject를 참조하세요.

`CreateBucket`

다음 코드 예시는 `CreateBucket`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

버킷을 만듭니다.

```
   /**
    * Creates an S3 bucket asynchronously.
    *
    * @param bucketName the name of the S3 bucket to create
    * @return a {@link CompletableFuture} that completes when the bucket is created
and ready
    * @throws RuntimeException if there is a failure while creating the bucket
    */
   public CompletableFuture<Void> createBucketAsync(String bucketName) {
       CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
           .bucket(bucketName)
           .build();

       CompletableFuture<CreateBucketResponse> response =
getAsyncClient().createBucket(bucketRequest);
       return response.thenCompose(resp -> {
           S3AsyncWaiter s3Waiter = getAsyncClient().waiter();
           HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
               .bucket(bucketName)
               .build();

           CompletableFuture<WaiterResponse<HeadBucketResponse>>
waiterResponseFuture =
```

```
                s3Waiter.waitUntilBucketExists(bucketRequestWait);
            return waiterResponseFuture.thenAccept(waiterResponse -> {
                waiterResponse.matched().response().ifPresent(headBucketResponse ->
{
                    logger.info(bucketName + " is ready");
                });
            });
        }).whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to create bucket", ex);
            }
        });
    }
```

객체 잠금을 활성화한 버킷을 생성합니다.

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

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateBucket을 참조하세요.

`DeleteBucket`

다음 코드 예시는 `DeleteBucket`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Deletes an S3 bucket asynchronously.
    *
    * @param bucket the name of the bucket to be deleted
    * @return a {@link CompletableFuture} that completes when the bucket deletion
is successful, or throws a {@link RuntimeException}
    * if an error occurs during the deletion process
    */
   public CompletableFuture<Void> deleteBucketAsync(String bucket) {
       DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
           .bucket(bucket)
           .build();

       CompletableFuture<DeleteBucketResponse> response =
getAsyncClient().deleteBucket(deleteBucketRequest);
       response.whenComplete((deleteRes, ex) -> {
           if (deleteRes != null) {
               logger.info(bucket + " was deleted.");
           } else {
               throw new RuntimeException("An S3 exception occurred during bucket
deletion", ex);
           }
       });
       return response.thenApply(r -> null);
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteBucket을 참조하세요.

`DeleteBucketPolicy`

다음 코드 예시는 `DeleteBucketPolicy`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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();
    }

    /**
     * Deletes the S3 bucket policy for the specified bucket.
     *
     * @param s3 the {@link S3Client} instance to use for the operation
     * @param bucketName the name of the S3 bucket for which the policy should be
 deleted
     *
     * @throws S3Exception if there is an error deleting 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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteBucketPolicy를 참조하세요.

`DeleteBucketWebsite`

다음 코드 예시는 `DeleteBucketWebsite`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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();
    }

    /**
     * Deletes the website configuration for an Amazon S3 bucket.
     *
     * @param s3 The {@link S3Client} instance used to interact with Amazon S3.
     * @param bucketName The name of the S3 bucket for which the website
 configuration should be deleted.
     * @throws S3Exception If an error occurs while deleting the website
 configuration.
     */
    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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteBucketWebsite를 참조하세요.

`DeleteObject`

다음 코드 예시는 `DeleteObject`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Deletes an object from an S3 bucket asynchronously.
    *
    * @param bucketName the name of the S3 bucket
    * @param key        the key (file name) of the object to be deleted
    * @return a {@link CompletableFuture} that completes when the object has been
deleted
    */
   public CompletableFuture<Void> deleteObjectFromBucketAsync(String bucketName,
String key) {
       DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
           .bucket(bucketName)
           .key(key)
           .build();

       CompletableFuture<DeleteObjectResponse> response =
getAsyncClient().deleteObject(deleteObjectRequest);
       response.whenComplete((deleteRes, ex) -> {
           if (deleteRes != null) {
               logger.info(key + " was deleted");
           } else {
               throw new RuntimeException("An S3 exception occurred during delete",
ex);
           }
       });

       return response.thenApply(r -> null);
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteObject를 참조하세요.

DeleteObjects

다음 코드 예시는 DeleteObjects의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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();
    }

    /**
     * Deletes multiple objects from an Amazon S3 bucket.
     *
     * @param s3 An Amazon S3 client object.
     * @param bucketName The name of the Amazon S3 bucket to delete objects from.
     */
    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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteObjects를 참조하세요.

`GetBucketAcl`

다음 코드 예시는 `GetBucketAcl`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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!");
```

```
    }

    /**
     * Retrieves the Access Control List (ACL) for an object in an Amazon S3 bucket.
     *
     * @param s3 The S3Client object used to interact with the Amazon S3 service.
     * @param objectKey The key of the object for which the ACL is to be retrieved.
     * @param bucketName The name of the bucket containing the object.
     * @return The ID of the grantee who has permission on the object, or an empty
 string if an error occurs.
     */
    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 "";
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetBucketAcl을 참조하세요.

GetBucketPolicy

다음 코드 예시는 GetBucketPolicy의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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();
    }

    /**
     * Retrieves the policy for the specified Amazon S3 bucket.
     *
     * @param s3 the {@link S3Client} instance to use for making the request
     * @param bucketName the name of the S3 bucket for which to retrieve the policy
     * @return the policy text for the specified bucket, or an empty string if an
 error occurs
     */
    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 "";
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetBucketPolicy를 참조하세요.

GetBucketReplication

다음 코드 예시는 GetBucketReplication의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Retrieves the replication details for the specified S3 bucket.
    *
    * @param s3Client           the S3 client used to interact with the S3 service
    * @param sourceBucketName   the name of the S3 bucket to retrieve the
replication details for
    *
    * @throws S3Exception if there is an error retrieving the replication details
    */
   public static void getReplicationDetails(S3Client s3Client, String
sourceBucketName) {
       GetBucketReplicationRequest getRequest =
GetBucketReplicationRequest.builder()
           .bucket(sourceBucketName)
           .build();

       try {
           ReplicationConfiguration replicationConfig =
s3Client.getBucketReplication(getRequest).replicationConfiguration();
           ReplicationRule rule = replicationConfig.rules().get(0);
           System.out.println("Retrieved destination bucket: " +
rule.destination().bucket());
           System.out.println("Retrieved priority: " + rule.priority());
           System.out.println("Retrieved source-bucket replication rule status: " +
rule.status());

       } catch (S3Exception e) {
           System.err.println("Failed to retrieve replication details: " +
e.awsErrorDetails().errorMessage());
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetBucketReplication을 참조하세요.

`GetObject`

다음 코드 예시는 `GetObject`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

S3Client를 사용하여 바이트 배열로 데이터를 읽습니다.

```
   /**
    * Asynchronously retrieves the bytes of an object from an Amazon S3 bucket and
writes them to a local file.
    *
    * @param bucketName the name of the S3 bucket containing the object
    * @param keyName    the key (or name) of the S3 object to retrieve
    * @param path       the local file path where the object's bytes will be
written
    * @return a {@link CompletableFuture} that completes when the object bytes have
been written to the local file
    */
   public CompletableFuture<Void> getObjectBytesAsync(String bucketName, String
keyName, String path) {
       GetObjectRequest objectRequest = GetObjectRequest.builder()
           .key(keyName)
           .bucket(bucketName)
           .build();

       CompletableFuture<ResponseBytes<GetObjectResponse>> response =
getAsyncClient().getObject(objectRequest, AsyncResponseTransformer.toBytes());
       return response.thenAccept(objectBytes -> {
           try {
               byte[] data = objectBytes.asByteArray();
```

```
                Path filePath = Paths.get(path);
                Files.write(filePath, data);
                logger.info("Successfully obtained bytes from an S3 object");
            } catch (IOException ex) {
                throw new RuntimeException("Failed to write data to file", ex);
            }
        }).whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to get object bytes from S3",
ex);
            }
        });
    }
```

S3TransferManager를 사용하여 S3 버킷 내의 객체를 로컬 파일로 다운로드합니다. 파일 전체를 보고 테스트합니다.

```
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.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.NoSuchFileException;
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();
    }
```

S3Client를 사용하여 객체에 속하는 태그를 읽습니다.

```
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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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();
    }

    /**
     * Lists the tags associated with an Amazon S3 object.
     *
     * @param s3 the S3Client object used to interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket that contains the object
     * @param keyName the key (name) of the S3 object
     */
    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);
```

```
        }
    }
}
```

S3Client를 사용하여 객체의 URL을 가져옵니다.

```
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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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();
    }

    /**
     * Retrieves the URL for a specific object in an Amazon S3 bucket.
     *
     * @param s3 the S3Client object used to interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket where the object is stored
     * @param keyName the name of the object for which the URL should be retrieved
     * @throws S3Exception if there is an error retrieving the URL for the specified
 object
     */
    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);
        }
    }
}
```

S3Client를 사용한 S3Presigner 클라이언트 객체를 사용하여 객체를 가져옵니다.

```
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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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();
```

```
    }

    /**
     * Generates a pre-signed URL for an Amazon S3 object.
     *
     * @param presigner The {@link S3Presigner} instance to use for generating the
pre-signed URL.
     * @param bucketName The name of the Amazon S3 bucket where the object is
stored.
     * @param keyName The key name (file name) of the object in the Amazon S3
bucket.
     *
     * @throws S3Exception If there is an error interacting with the Amazon S3
service.
     * @throws IOException If there is an error opening the HTTP connection or
reading/writing the request/response.
     */
    public static void getPresignedUrl(S3Presigner presigner, String bucketName,
String keyName) {
        try {
            GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            GetObjectPresignRequest getObjectPresignRequest =
GetObjectPresignRequest.builder()
                .signatureDuration(Duration.ofMinutes(60))
                .getObjectRequest(getObjectRequest)
                .build();

            PresignedGetObjectRequest presignedGetObjectRequest =
presigner.presignGetObject(getObjectPresignRequest);
            String theUrl = presignedGetObjectRequest.url().toString();
            System.out.println("Presigned URL: " + theUrl);
            HttpURLConnection connection = (HttpURLConnection)
presignedGetObjectRequest.url().openConnection();
            presignedGetObjectRequest.httpRequest().headers().forEach((header,
values) -> {
                values.forEach(value -> {
                    connection.addRequestProperty(header, value);
                });
            });
```

```
            // Send any request payload that the service needs (not needed when
            // isBrowserExecutable is true).
            if (presignedGetObjectRequest.signedPayload().isPresent()) {
                connection.setDoOutput(true);

                try (InputStream signedPayload =
 presignedGetObjectRequest.signedPayload().get().asInputStream();
                     OutputStream httpOutputStream = connection.getOutputStream()) {
                    IoUtils.copy(signedPayload, httpOutputStream);
                }
            }

            // Download the result of executing the request.
            try (InputStream content = connection.getInputStream()) {
                System.out.println("Service returned response: ");
                IoUtils.copy(content, System.out);
            }

        } catch (S3Exception | IOException e) {
            e.printStackTrace();
        }
    }
}
```

ResponseTransformer 객체와 S3Client를 사용하여 객체를 가져옵니다.

```
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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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);
        s3.close();
    }

    /**
     * Retrieves the bytes of an object stored in an Amazon S3 bucket and saves them
 to a local file.
     *
     * @param s3 The S3Client instance used to interact with the Amazon S3 service.
     * @param bucketName The name of the S3 bucket where the object is stored.
     * @param keyName The key (or name) of the S3 object.
```

```
     * @param path The local file path where the object's bytes will be saved.
     * @throws IOException If an I/O error occurs while writing the bytes to the
 local file.
     * @throws S3Exception If an error occurs while retrieving the object from the
 S3 bucket.
     */
    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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetObject를 참조하세요.

GetObjectLegalHold

다음 코드 예시는 GetObjectLegalHold의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   // 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;
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetObjectLegalHold를 참조하세요.

`GetObjectLockConfiguration`

다음 코드 예시는 `GetObjectLockConfiguration`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   // 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());
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetObjectLockConfiguration을 참조하세요.

`GetObjectRetention`

다음 코드 예시는 `GetObjectRetention`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   // 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;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetObjectRetention을 참조하세요.

`HeadObject`

다음 코드 예시는 `HeadObject`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

객체의 콘텐츠 유형을 결정합니다.

```
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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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();
    }

    /**
     * Retrieves the content type of an object stored in an Amazon S3 bucket.
     *
     * @param s3 an instance of the {@link S3Client} class, which is used to
 interact with the Amazon S3 service
```

```
     * @param bucketName the name of the S3 bucket where the object is stored
     * @param keyName the key (file name) of the object in the S3 bucket
     */
    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);
        }
    }
}
```

객체의 복원 상태를 가져옵니다.

```
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();
    }

    /**
     * Checks the restoration status of an Amazon S3 object.
     *
     * @param s3         an instance of the {@link S3Client} class used to interact
 with the Amazon S3 service
     * @param bucketName the name of the Amazon S3 bucket where the object is stored
     * @param keyName    the name of the Amazon S3 object to be checked
     * @throws S3Exception if an error occurs while interacting with the Amazon S3
 service
     */
    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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 HeadObject를 참조하세요.

`ListBuckets`

다음 코드 예시는 `ListBuckets`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListBucketsIterable;
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * 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);

    }

    /**
     * Lists all the S3 buckets available in the current AWS account.
     *
```

```
     * @param s3 The {@link S3Client} instance to use for interacting with the
 Amazon S3 service.
     */
    public static void listAllBuckets(S3Client s3) {
        ListBucketsIterable response = s3.listBucketsPaginator();
        response.buckets().forEach(bucket ->
            System.out.println("Bucket Name: " + bucket.name()));
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListBuckets를 참조하세요.

`ListMultipartUploads`

다음 코드 예시는 `ListMultipartUploads`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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();
    }

    /**
     * Lists the multipart uploads currently in progress in the specified Amazon S3
 bucket.
     *
     * @param s3 the S3Client object used to interact with Amazon S3
     * @param bucketName the name of the Amazon S3 bucket to list the multipart
 uploads for
     */
    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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListMultipartUploads를 참조하세요.

`ListObjectsV2`

다음 코드 예시는 `ListObjectsV2`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Asynchronously lists all objects in the specified S3 bucket.
     *
     * @param bucketName the name of the S3 bucket to list objects for
     * @return a {@link CompletableFuture} that completes when all objects have been
 listed
     */
    public CompletableFuture<Void> listAllObjectsAsync(String bucketName) {
        ListObjectsV2Request initialRequest = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .maxKeys(1)
            .build();

        ListObjectsV2Publisher paginator =
 getAsyncClient().listObjectsV2Paginator(initialRequest);
```

```
        return paginator.subscribe(response -> {
            response.contents().forEach(s3Object -> {
                logger.info("Object key: " + s3Object.key());
            });
        }).thenRun(() -> {
            logger.info("Successfully listed all objects in the bucket: " +
 bucketName);
        }).exceptionally(ex -> {
            throw new RuntimeException("Failed to list objects", ex);
        });
    }
```

페이지 매김을 사용하여 객체를 나열합니다.

```
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();
    }

    /**
     * Lists the objects in the specified S3 bucket.
     *
     * @param s3 the S3Client instance used to interact with Amazon S3
     * @param bucketName the name of the S3 bucket to list the objects from
     */
    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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListObjectsV2를 참조하세요.

PutBucketAcl

다음 코드 예시는 PutBucketAcl의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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);
            return;
        }

        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();
    }

    /**
     * Sets the Access Control List (ACL) for an Amazon S3 bucket.
     *
     * @param s3 the S3Client instance to be used for the operation
     * @param bucketName the name of the S3 bucket to set the ACL for
     * @param id the ID of the AWS user or account that will be granted full control
of the bucket
     * @throws S3Exception if an error occurs while setting the bucket ACL
     */
    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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutBucketAcl을 참조하세요.

`PutBucketCors`

다음 코드 예시는 `PutBucketCors`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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();
    }

    /**
     * Deletes the CORS (Cross-Origin Resource Sharing) configuration for an Amazon
 S3 bucket.
     *
```

```
    * @param s3            the {@link S3Client} instance used to interact with the
Amazon S3 service
    * @param bucketName    the name of the Amazon S3 bucket for which the CORS
configuration should be deleted
    * @param accountId     the expected AWS account ID of the bucket owner
    *
    * @throws S3Exception if an error occurs while deleting the CORS configuration
for the bucket
    */
   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);
       }
   }

   /**
    * Retrieves the CORS (Cross-Origin Resource Sharing) configuration for the
specified S3 bucket.
    *
    * @param s3 the S3Client instance to use for the operation
    * @param bucketName the name of the S3 bucket to retrieve the CORS
configuration for
    * @param accountId the expected bucket owner's account ID
    *
    * @throws S3Exception if there is an error retrieving the CORS configuration
    */
   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);
        }
    }

    /**
     * Sets the Cross-Origin Resource Sharing (CORS) rules for an Amazon S3 bucket.
     *
     * @param s3 The S3Client object used to interact with the Amazon S3 service.
     * @param bucketName The name of the S3 bucket to set the CORS rules for.
     * @param accountId The AWS account ID of the bucket owner.
     */
    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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutBucketCors를 참조하세요.

`PutBucketLifecycleConfiguration`

다음 코드 예시는 `PutBucketLifecycleConfiguration`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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();
    }

    /**
     * Sets the lifecycle configuration for an Amazon S3 bucket.
     *
     * @param s3           The Amazon S3 client to use for the operation.
     * @param bucketName   The name of the Amazon S3 bucket.
     * @param accountId    The expected owner of the Amazon S3 bucket.
     *
     * @throws S3Exception if there is an error setting the lifecycle configuration.
     */
    public static void setLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
        try {
            // Create a rule to archive objects with the "glacierobjects/" prefix to
the
            // S3 Glacier Flexible Retrieval storage class immediately.
            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);
        }
    }

    /**
```

```
    * Retrieves the lifecycle configuration for an Amazon S3 bucket and adds a new
lifecycle rule to it.
    *
    * @param s3 the S3Client instance used to interact with Amazon S3
    * @param bucketName the name of the Amazon S3 bucket
    * @param accountId the expected owner of the Amazon S3 bucket
    */
   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);
        }
    }

    /**
     * Deletes the lifecycle configuration for an Amazon S3 bucket.
     *
     * @param s3 the {@link S3Client} to use for the operation
     * @param bucketName the name of the S3 bucket
     * @param accountId the expected account owner of the S3 bucket
     *
     * @throws S3Exception if an error occurs while deleting the lifecycle
configuration
     */
    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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutBucketLifecycleConfiguration을 참조하세요.

`PutBucketPolicy`

다음 코드 예시는 `PutBucketPolicy`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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();
    }

    /**
     * Sets the policy for an Amazon S3 bucket.
     *
     * @param s3         the {@link S3Client} object used to interact with the
 Amazon S3 service
     * @param bucketName the name of the Amazon S3 bucket
     * @param policyText the text of the policy to be set on the bucket
     * @throws S3Exception if there is an error setting the bucket policy
```

```
     */
    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!");
    }

    /**
     * Retrieves the bucket policy from a specified file.
     *
     * @param policyFile the path to the file containing the bucket policy
     * @return the content of the bucket policy file as a string
     */
    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();
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutBucketPolicy를 참조하세요.

`PutBucketReplication`

다음 코드 예시는 `PutBucketReplication`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Sets the replication configuration for an Amazon S3 bucket.
     *
     * @param s3Client             the S3Client instance to use for the operation
     * @param sourceBucketName     the name of the source bucket
     * @param destBucketName       the name of the destination bucket
     * @param destinationBucketARN the Amazon Resource Name (ARN) of the destination
 bucket
     * @param roleARN              the ARN of the IAM role to use for the
 replication configuration
     */
    public static void setReplication(S3Client s3Client, String sourceBucketName,
 String destBucketName, String destinationBucketARN, String roleARN) {
```

```
        try {
            Destination destination = Destination.builder()
                .bucket(destinationBucketARN)
                .storageClass(StorageClass.STANDARD)
                .build();

            // Define a prefix filter for replication.
            ReplicationRuleFilter ruleFilter = ReplicationRuleFilter.builder()
                .prefix("documents/")
                .build();

            // Define delete marker replication setting.
            DeleteMarkerReplication deleteMarkerReplication =
DeleteMarkerReplication.builder()
                .status(DeleteMarkerReplicationStatus.DISABLED)
                .build();

            // Create the replication rule.
            ReplicationRule replicationRule = ReplicationRule.builder()
                .priority(1)
                .filter(ruleFilter)
                .status(ReplicationRuleStatus.ENABLED)
                .deleteMarkerReplication(deleteMarkerReplication)
                .destination(destination)
                .build();

            List<ReplicationRule> replicationRuleList = new ArrayList<>();
            replicationRuleList.add(replicationRule);

            // Define the replication configuration with IAM role.
            ReplicationConfiguration configuration =
ReplicationConfiguration.builder()
                .role(roleARN)
                .rules(replicationRuleList)
                .build();

            // Apply the replication configuration to the source bucket.
            PutBucketReplicationRequest replicationRequest =
PutBucketReplicationRequest.builder()
                .bucket(sourceBucketName)
                .replicationConfiguration(configuration)
                .build();

            s3Client.putBucketReplication(replicationRequest);
```

```
            System.out.println("Replication configuration set successfully.");

        } catch (IllegalArgumentException e) {
            System.err.println("Configuration error: " + e.getMessage());
        } catch (S3Exception e) {
            System.err.println("S3 Exception: " +
e.awsErrorDetails().errorMessage());
            System.err.println("Status Code: " + e.statusCode());
            System.err.println("Error Code: " + e.awsErrorDetails().errorCode());


        } catch (SdkException e) {
            System.err.println("SDK Exception: " + e.getMessage());
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutBucketReplication을 참조하세요.

`PutBucketVersioning`

다음 코드 예시는 `PutBucketVersioning`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Enables bucket versioning for the specified S3 bucket.
     *
     * @param s3Client the S3 client to use for the operation
     * @param bucketName the name of the S3 bucket to enable versioning for
     */
    public static void enableBucketVersioning(S3Client s3Client, String bucketName){
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .status(BucketVersioningStatus.ENABLED)
            .build();
```

```
        PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        s3Client.putBucketVersioning(versioningRequest);
        System.out.println("Bucket versioning has been enabled for "+bucketName);
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutBucketVersioning을 참조하세요.

`PutBucketWebsite`

다음 코드 예시는 `PutBucketWebsite`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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();
    }

    /**
     * Sets the website configuration for an Amazon S3 bucket.
     *
     * @param s3 The {@link S3Client} instance to use for the AWS SDK operations.
     * @param bucketName The name of the S3 bucket to configure.
     * @param indexDoc The name of the index document to use for the website
 configuration.
     */
    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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutBucketWebsite를 참조하세요.

`PutObject`

다음 코드 예시는 `PutObject`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

S3Client를 사용하여 버킷에 파일을 업로드합니다.

```
    /**
     * Uploads a local file to an AWS S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to upload the file to
     * @param key        the key (object name) to use for the uploaded file
     * @param objectPath the local file path of the file to be uploaded
```

```
     * @return a {@link CompletableFuture} that completes with the {@link
 PutObjectResponse} when the upload is successful, or throws a {@link
 RuntimeException} if the upload fails
     */
    public CompletableFuture<PutObjectResponse> uploadLocalFileAsync(String
 bucketName, String key, String objectPath) {
        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        CompletableFuture<PutObjectResponse> response =
 getAsyncClient().putObject(objectRequest,
 AsyncRequestBody.fromFile(Paths.get(objectPath)));
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to upload file", ex);
            }
        });
    }
```

S3TransferManager를 사용하여 버킷에 파일을 업로드합니다. 파일 전체를 보고 테스트합니다.

```
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();
    }
```

S3Client를 사용하여 버킷에 객체를 업로드하고 태그를 설정합니다.

```
    /**
     * Puts tags on an Amazon S3 object.
     *
     * @param s3 An {@link S3Client} object that represents the Amazon S3 client.
     * @param bucketName The name of the Amazon S3 bucket.
     * @param objectKey The key of the Amazon S3 object.
     * @param objectPath The file path of the object to be uploaded.
     */
    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);
        }
    }

    /**
     * Updates the tags associated with an object in an Amazon S3 bucket.
     *
     * @param s3 an instance of the S3Client class, which is used to interact with
the Amazon S3 service
     * @param bucketName the name of the S3 bucket containing the object
     * @param objectKey the key (or name) of the object in the S3 bucket
     * @throws S3Exception if there is an error updating the object's tags
     */
    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);
        }
    }

    /**
     * Retrieves the contents of a file as a byte array.
     *
     * @param filePath the path of the file to be read
     * @return a byte array containing the contents of the file, or null if an error
occurs
     */
    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;
    }
}
```

S3Client를 사용하여 버킷에 객체를 업로드하고 메타데이터를 설정합니다.

```
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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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();
    }

    /**
     * Uploads an object to an Amazon S3 bucket with metadata.
     *
     * @param s3 the S3Client object used to interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket to upload the object to
     * @param objectKey the name of the object to be uploaded
     * @param objectPath the local file path of the object to be uploaded
     */
```

```
    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);
        }
    }
}
```

S3Client를 사용하여 버킷에 객체를 업로드하고 객체 보존 값을 설정합니다.

```
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.
 * <p>
```

```
 * For more information, see the following documentation topic:
 * <p>
 * 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();
    }

    /**
     * Sets the retention period for an object in an Amazon S3 bucket.
     *
     * @param s3     the S3Client object used to interact with the Amazon S3 service
     * @param key    the key (name) of the object in the S3 bucket
     * @param bucket the name of the S3 bucket where the object is stored
     *
     * @throws S3Exception if an error occurs while setting the object retention
 period
     */
```

```
    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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutObject를 참조하세요.

PutObjectLegalHold

다음 코드 예시는 PutObjectLegalHold의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   // 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
+".");
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutObjectLegalHold를 참조하세요.

`PutObjectLockConfiguration`

다음 코드 예시는 `PutObjectLockConfiguration`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

버킷의 객체 잠금 구성을 설정합니다.

```
   // 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() +
"'");
       }
   }
```

버킷의 기본 보존 기간을 설정합니다.

```
   // 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 +".");
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutObjectLockConfiguration을 참조하세요.

`PutObjectRetention`

다음 코드 예시는 `PutObjectRetention`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    // 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 +".");
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutObjectRetention을 참조하세요.

`RestoreObject`

다음 코드 예시는 `RestoreObject`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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();
    }

    /**
     * Restores an S3 object from the Glacier storage class.
     *
     * @param s3                   an instance of the {@link S3Client} to be used
 for interacting with Amazon S3
     * @param bucketName           the name of the S3 bucket where the object is
 stored
     * @param keyName              the key (object name) of the S3 object to be
 restored
     * @param expectedBucketOwner  the AWS account ID of the expected bucket owner
```

```
     */
    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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 RestoreObject를 참조하세요.

`SelectObjectContent`

다음 코드 예시는 `SelectObjectContent`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

다음 예시에서는 JSON 객체를 사용한 쿼리를 보여줍니다. 전체 예시는 CSV 객체의 사용도 보여줍니다.

```
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 = "amzn-s3-demo-bucket-" + 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;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 SelectObjectContent를 참조하세요.

`UploadPartCopy`

다음 코드 예시는 `UploadPartCopy`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   public CompletableFuture<String> performMultiCopy(String toBucket, String
bucketName, String key) {
        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
            .bucket(toBucket)
            .key(key)
            .build();

        getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
            .thenApply(createMultipartUploadResponse -> {
                String uploadId = createMultipartUploadResponse.uploadId();
                System.out.println("Upload ID: " + uploadId);

                UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
                    .sourceBucket(bucketName)
```

```
                    .destinationBucket(toBucket)
                    .sourceKey(key)
                    .destinationKey(key)
                    .uploadId(uploadId)  // Use the valid uploadId.
                    .partNumber(1)  // Ensure the part number is correct.
                    .copySourceRange("bytes=0-1023")  // Adjust range as needed
                    .build();

                return getAsyncClient().uploadPartCopy(uploadPartCopyRequest);
            })
            .thenCompose(uploadPartCopyFuture -> uploadPartCopyFuture)
            .whenComplete((uploadPartCopyResponse, exception) -> {
                if (exception != null) {
                    // Handle any exceptions.
                    logger.error("Error during upload part copy: " +
exception.getMessage());
                } else {
                    // Successfully completed the upload part copy.
                    System.out.println("Upload Part Copy completed successfully.
ETag: " + uploadPartCopyResponse.copyPartResult().eTag());
                }
            });
        return null;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 UploadPartCopy를 참조하세요.

시나리오

버킷이 존재하는지 확인

다음 코드 예제에서는 버킷이 존재하는지 확인하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

다음 doesBucketExists 메서드로 SDK for Java V1 AmazonS3Client#doesBucketExistV2(String) 메서드를 대체할 수 있습니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.http.HttpStatusCode;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.utils.Validate;

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

    public static void main(String[] args) {
        DoesBucketExist doesBucketExist = new DoesBucketExist();

        final S3Client s3SyncClient = S3Client.builder().build();
        final String bucketName = "amzn-s3-demo-bucket"; // Change to the bucket
 name that you want to check.

        boolean exists = doesBucketExist.doesBucketExist(bucketName, s3SyncClient);
        logger.info("Bucket exists: {}", exists);
    }

    /**
     * Checks if the specified bucket exists. Amazon S3 buckets are named in a
 global namespace; use this method to
     * determine if a specified bucket name already exists, and therefore can't be
 used to create a new bucket.
     * <p>
     * Internally this method uses the <a
     * href="https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/
services/s3/
S3Client.html#getBucketAcl(java.util.function.Consumer)">S3Client.getBucketAcl(String)</
a>
     * operation to determine whether the bucket exists.
     * <p>
     * This method is equivalent to the AWS SDK for Java V1's <a
     * href="https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/
com/amazonaws/services/s3/AmazonS3Client.html#doesBucketExistV2-
java.lang.String-">AmazonS3Client#doesBucketExistV2(String)</a>.
```

```
     *
     * @param bucketName   The name of the bucket to check.
     * @param s3SyncClient An <code>S3Client</code> instance. The method checks for
 the bucket in the AWS Region
     *                     configured on the instance.
     * @return The value true if the specified bucket exists in Amazon S3; the value
 false if there is no bucket in
     *         Amazon S3 with that name.
     */
    public boolean doesBucketExist(String bucketName, S3Client s3SyncClient) {
        try {
            Validate.notEmpty(bucketName, "The bucket name must not be null or an
 empty string.", "");
            s3SyncClient.getBucketAcl(r -> r.bucket(bucketName));
            return true;
        } catch (AwsServiceException ase) {
            // A redirect error or an AccessDenied exception means the bucket exists
 but it's not in this region
            // or we don't have permissions to it.
            if ((ase.statusCode() == HttpStatusCode.MOVED_PERMANENTLY) ||
 "AccessDenied".equals(ase.awsErrorDetails().errorCode())) {
                return true;
            }
            if (ase.statusCode() == HttpStatusCode.NOT_FOUND) {
                return false;
            }
            throw ase;
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetBucketAcl을 참조하세요.

미리 서명된 URL 생성

다음 코드 예제에서는 Amazon S3에 대해 미리 서명된 URL을 생성하고 객체를 업로드하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

다음은 미리 서명된 URL을 만들고 HTTP 클라이언트 라이브러리를 이용해 해당 URL을 사용하는 방법의 세 가지 예제를 보여줍니다.

- 3개의 HTTP 클라이언트 라이브러리를 이용해 URL을 사용하는 HTTP GET 요청
- 헤더에 메타데이터가 포함되어 있고 3개의 HTTP 클라이언트 라이브러리를 이용해 URL을 사용하는 HTTP PUT 요청
- 쿼리 파라미터가 있고 하나의 HTTP 클라이언트 라이브러리를 이용해 URL을 사용하는 HTTP PUT 요청

객체에 대해 미리 서명된 URL을 생성한 다음 다운로드(GET 요청)합니다.

가져옵니다.

```
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;
```

URL을 생성합니다.

```
    /* 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();
        }
    }
```

다음 세 가지 방법 중 하나를 사용하여 객체를 다운로드합니다.

JDK `HttpURLConnection`(v1.1 이후) 클래스를 사용하여 다운로드합니다.

```
   /* 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();
   }
```

JDK `HttpClient`(v11 이후) 클래스를 사용하여 다운로드합니다.

```
   /* 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();
    }
```

AWS SDK for Java SdkHttpClient 클래스를 사용하여 다운로드를 수행합니다.

```
    /* Use the AWS SDK for Java SdkHttpClient class to do the download. */
    public byte[] useSdkHttpClientToGet(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();
    }
```

업로드를 위해 헤더에 메타데이터가 있는 미리 서명된 URL을 생성한 다음 파일을 업로드(PUT 요청)합니다.

가져옵니다.

```
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;
```

URL을 생성합니다.

```
   /* 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();
       }
   }
```

다음 세 가지 방법 중 하나를 사용하여 파일 객체를 업로드합니다.

JDK `HttpURLConnection`(v1.1 이후) 클래스를 사용하여 업로드합니다.

```
   /* 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);
       }
   }
```

JDK `HttpClient`(v11 이후) 클래스를 사용하여 업로드합니다.

```
   /* 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);
        }
    }
```

AWS for Java V2 SdkHttpClient 클래스를 사용하여 업로드를 수행합니다.

```
    /* 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);
        }
    }
```

업로드를 위해 쿼리 파라미터가 있는 미리 서명된 URL을 생성한 다음 파일을 업로드(PUT 요청)합니다.

가져옵니다.

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.awscore.AwsRequestOverrideConfiguration;
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.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.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.time.Duration;
```

```
import java.util.Map;
import java.util.UUID;
```

URL을 생성합니다.

```
   /**
    *  Creates a presigned URL to use in a subsequent HTTP PUT request. The code
adds query parameters
    *  to the request instead of using headers. By using query parameters, you do
not need to add the
    *  the parameters as headers when the PUT request is eventually sent.
    *
    * @param bucketName Bucket name where the object will be uploaded.
    * @param keyName Key name of the object that will be uploaded.
    * @param queryParams Query string parameters to be added to the presigned URL.
    * @return
    */
   public String createPresignedUrl(String bucketName, String keyName, Map<String,
String> queryParams) {
       try (S3Presigner presigner = S3Presigner.create()) {
           // Create an override configuration to store the query parameters.
           AwsRequestOverrideConfiguration.Builder overrideConfigurationBuilder =
AwsRequestOverrideConfiguration.builder();

           queryParams.forEach(overrideConfigurationBuilder::putRawQueryParameter);

           PutObjectRequest objectRequest = PutObjectRequest.builder()
                   .bucket(bucketName)
                   .key(keyName)
                   .overrideConfiguration(overrideConfigurationBuilder.build()) //
Add the override configuration.
                   .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();
        }
    }
```

AWS for Java V2 SdkHttpClient 클래스를 사용하여 업로드를 수행합니다.

```
    /**
     * Use the AWS SDK for Java V2 SdkHttpClient class to execute the PUT request.
Since the
     * URL contains the query parameters, no headers are needed for metadata, SSE
settings, or ACL settings.
     *
     * @param presignedUrlString The URL for the PUT request.
     * @param fileToPut File to uplaod
     */
    public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut) {
        logger.info("Begin [{}] upload", fileToPut.toString());

        try {
            URL presignedUrl = new URL(presignedUrlString);

            SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
                    .method(SdkHttpMethod.PUT)
                    .uri(presignedUrl.toURI());

            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);
        }
    }
```

사진을 관리하기 위한 서버리스 애플리케이션 만들기

다음 코드 예시에서는 사용자가 레이블을 사용하여 사진을 관리할 수 있는 서버리스 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 AWS 커뮤니티의 게시물을 참조하세요.

이 예제에서 사용되는 서비스

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

미완료 멀티파트 업로드 삭제

다음 코드 예시는 미완료 Amazon S3 멀티파트 업로드를 삭제 또는 중지하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

어떤 이유로든 진행 중이거나 완료되지 않은 멀티파트 업로드를 중지하려면 다음 예와 같이 업로드 목록을 가져온 다음 삭제하면 됩니다.

```
   /**
    * Aborts all incomplete multipart uploads from the specified S3 bucket.
    * <p>
    * This method retrieves a list of all incomplete multipart uploads in the
specified S3 bucket,
    * and then aborts each of those uploads.
    */
   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);
```

```
            }
        }
    }
```

특정 날짜 전 또는 후에 시작된 미완료 멀티파트 업로드를 삭제하려면 다음 예와 같이 특정 시점을 기준으로 멀티파트 업로드를 선택적으로 삭제할 수 있습니다.

```
    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);
                }
            }
        }
    }
```

멀티파트 업로드를 시작한 후 업로드 ID에 액세스할 수 있는 경우 ID를 사용하여 진행 중인 업로드를 삭제할 수 있습니다.

```
   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);
       }
   }
```

특정 일수보다 오래된 미완료 멀티파트 업로드를 지속적으로 삭제하려면 버킷의 버킷 수명 주기 구성을 설정하세요. 다음 예시에서는 7일보다 오래된 미완료 업로드를 삭제하는 규칙을 생성하는 방법을 보여줍니다.

```
   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());
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - AbortMultipartUpload
  - ListMultipartUploads
  - PutBucketLifecycleConfiguration

이미지에서 PPE 감지

다음 코드 예제에서는 Amazon Rekognition을 사용하여 이미지에서 개인 보호 장비(PPE)를 감지하는 앱을 구축하는 방법을 보여줍니다.

SDK for Java 2.x

개인 보호 장비로 이미지를 감지하는 AWS Lambda 함수를 생성하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

이미지에서 객체 감지

다음 코드 예제에서는 Amazon Rekognition을 사용하여 이미지에서 범주별 객체를 감지하는 앱을 구축하는 방법을 보여줍니다.

SDK for Java 2.x

Amazon Rekognition을 사용하여 Amazon Simple Storage Service(Amazon S3) 버킷에 있는 이미지에서 범주별로 객체를 식별하기 위해 Amazon Rekognition을 사용하여 앱을 만드는 방법을 보여줍니다. 이 앱은 Amazon Simple Email Service(Amazon SES)를 사용하여 결과와 함께 이메일 알림을 관리자에게 보냅니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Rekognition

- Amazon S3
- Amazon SES

동영상에서 사람과 객체 감지

다음 코드 예제에서는 Amazon Rekognition을 사용하여 동영상에서 사람과 객체를 감지하는 방법을 보여줍니다.

SDK for Java 2.x

Amazon Rekognition Java API를 사용하여 Amazon Simple Storage Service(Amazon S3) 버킷에 있는 동영상에서 얼굴과 객체를 감지하기 위한 앱을 만드는 방법을 보여줍니다. 이 앱은 Amazon Simple Email Service(Amazon SES)를 사용하여 결과와 함께 이메일 알림을 관리자에게 보냅니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

S3 '디렉터리' 다운로드

다음 코드 예제에서는 Amazon S3 버킷 '디렉터리'의 콘텐츠를 다운로드하고 필터링하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예제에서는에서 S3TransferManager를 사용하여 Amazon S3 버킷에서 '디렉터리'를 다운로드 AWS SDK for Java 2.x 하는 방법을 보여줍니다. 또한 요청에서 DownloadFilters를 사용하는 방법을 보여줍니다.

```
   /**
    * For standard buckets, S3 provides the illusion of a directory structure
through the use of keys. When you upload
    * an object to an S3 bucket, you specify a key, which is essentially the "path"
to the object. The key can contain
    * forward slashes ("/") to make it appear as if the object is stored in a
directory structure, but this is just a
    * logical representation, not an actual directory.
    * <p><pre>
    * In this example, our S3 bucket contains the following objects:
    *
    * folder1/file1.txt
    * folder1/file2.txt
    * folder1/file3.txt
    * folder2/file1.txt
    * folder2/file2.txt
    * folder2/file3.txt
    * folder3/file1.txt
    * folder3/file2.txt
    * folder3/file3.txt
    *
    * When method `downloadS3Directories` is invoked with
    * `destinationPathURI` set to `/test`, the downloaded
    * directory looks like:
    *
    * |- test
    *    |- folder1
    *       |- file1.txt
    *       |- file2.txt
    *       |- file3.txt
    *    |- folder3
    *       |- file1.txt
    *       |- file2.txt
    *       |- file3.txt
    * </pre>
    *
    * @param transferManager    An S3TransferManager instance.
    * @param destinationPathURI local directory to hold the downloaded S3
'directories' and files.
```

```
     * @param bucketName         The S3 bucket that contains the 'directories' to
 download.
     * @return The number of objects (files, in this case) that were downloaded.
     */
    public Integer downloadS3Directories(S3TransferManager transferManager,
                                         URI destinationPathURI, String bucketName)
 {

        // Define the filters for which 'directories' we want to download.
        DownloadFilter folder1Filter = (S3Object s3Object) ->
 s3Object.key().startsWith("folder1/");
        DownloadFilter folder3Filter = (S3Object s3Object) ->
 s3Object.key().startsWith("folder3/");
        DownloadFilter folderFilter = s3Object ->
 folder1Filter.or(folder3Filter).test(s3Object);

        DirectoryDownload directoryDownload =
 transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
                .destination(Paths.get(destinationPathURI))
                .bucket(bucketName)
                .filter(folderFilter)
                .build());
        CompletedDirectoryDownload completedDirectoryDownload =
 directoryDownload.completionFuture().join();

        Integer numFilesInFolder1 =
 Paths.get(destinationPathURI).resolve("folder1").toFile().list().length;
        Integer numFilesInFolder3 =
 Paths.get(destinationPathURI).resolve("folder3").toFile().list().length;

        try {
            assert numFilesInFolder1 == 3;
            assert numFilesInFolder3 == 3;
            assert !
Paths.get(destinationPathURI).resolve("folder2").toFile().exists(); // `folder2` was
 not downloaded.
        } catch (AssertionError e) {
            logger.error("An assertion failed.");
        }

        completedDirectoryDownload.failedTransfers()
                .forEach(fail -> logger.warn("Object failed to transfer  [{}]",
 fail.exception().getMessage()));
        return numFilesInFolder1 + numFilesInFolder3;
```

```
    }
```

로컬 디렉터리로 객체 다운로드

다음 코드 예제에서는 Amazon Simple Storage Service(Amazon S3) 버킷 내의 모든 객체를 로컬 디렉터리로 다운로드하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

S3TransferManager를 사용하여 동일한 S3 버킷 내에 있는 모든 S3 객체를 다운로드합니다. 파일 전체를 보고 테스트합니다.

```
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();
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DownloadDirectory를 참조하세요.

Amazon S3 객체 잠그기

다음 코드 예시에는 S3 객체 잠금 기능을 사용하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon S3 객체 잠금 기능을 시연하는 대화형 시나리오를 실행합니다.

```
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.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) {
        final String usage = """
            Usage:
                <bucketName> \s

            Where:
                bucketName - The Amazon S3 bucket name.
           """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
        s3LockActions = new S3LockActions();
        bucketName = args[0];
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Simple Storage Service (S3) Object
 Locking Feature 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;
    }
}
```

S3 함수의 래퍼 클래스입니다.

```
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());
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold
  - PutObjectLockConfiguration
  - PutObjectRetention

S3를 사용하여 대규모 메시지 관리

다음 코드 예제에서는 Amazon SQS 확장 클라이언트 라이브러리를 사용하여 대규모 Amazon SQS 메시지를 작업하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.joda.time.DateTime;
import org.joda.time.format.DateTimeFormat;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
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.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.LifecycleExpiration;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueResponse;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;

import java.util.Arrays;
import java.util.List;
import java.util.UUID;

/**
 * Example of using Amazon SQS Extended Client Library for Java 2.x.
 */
public class SqsExtendedClientExample {
    private static final Logger logger =
 LoggerFactory.getLogger(SqsExtendedClientExample.class);

    private String s3BucketName;
    private String queueUrl;
    private final String queueName;
    private final S3Client s3Client;
    private final SqsClient sqsExtendedClient;
    private final int messageSize;

    /**
```

```
     * Constructor with default clients and message size.
     */
    public SqsExtendedClientExample() {
        this(S3Client.create(), 300000);
    }

    /**
     * Constructor with custom S3 client and message size.
     *
     * @param s3Client The S3 client to use
     * @param messageSize The size of the test message to create
     */
    public SqsExtendedClientExample(S3Client s3Client, int messageSize) {
        this.s3Client = s3Client;
        this.messageSize = messageSize;

        // Generate a unique bucket name.
        this.s3BucketName = UUID.randomUUID() + "-" +
                DateTimeFormat.forPattern("yyMMdd-hhmmss").print(new DateTime());

        // Generate a unique queue name.
        this.queueName = "MyQueue-" + UUID.randomUUID();

        // Configure the SQS extended client.
        final ExtendedClientConfiguration extendedClientConfig = new
ExtendedClientConfiguration()
                .withPayloadSupportEnabled(s3Client, s3BucketName);

        this.sqsExtendedClient = new
AmazonSQSExtendedClient(SqsClient.builder().build(), extendedClientConfig);
    }

    public static void main(String[] args) {
        SqsExtendedClientExample example = new SqsExtendedClientExample();
        try {
            example.setup();
            example.sendAndReceiveMessage();
        } finally {
            example.cleanup();
        }
    }

    /**
     * Send a large message and receive it back.
```

```
     *
     * @return The received message
     */
    public Message sendAndReceiveMessage() {
        try {
            // Create a large message.
            char[] chars = new char[messageSize];
            Arrays.fill(chars, 'x');
            String largeMessage = new String(chars);

            // Send the message.
            final SendMessageRequest sendMessageRequest =
SendMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageBody(largeMessage)
                    .build();

            sqsExtendedClient.sendMessage(sendMessageRequest);
            logger.info("Sent message of size: {}", largeMessage.length());

            // Receive and return the message.
            final ReceiveMessageResponse receiveMessageResponse =
sqsExtendedClient.receiveMessage(
                    ReceiveMessageRequest.builder().queueUrl(queueUrl).build());

            List<Message> messages = receiveMessageResponse.messages();
            if (messages.isEmpty()) {
                throw new RuntimeException("No messages received");
            }

            Message message = messages.getFirst();
            logger.info("\nMessage received.");
            logger.info("  ID: {}", message.messageId());
            logger.info("  Receipt handle: {}", message.receiptHandle());
            logger.info("  Message body size: {}", message.body().length());
            logger.info("  Message body (first 5 characters): {}",
message.body().substring(0, 5));

            return message;
        } catch (RuntimeException e) {
            logger.error("Error during message processing: {}", e.getMessage(), e);
            throw e;
        }
    }
```

- 자세한 정보는 AWS SDK for Java 2.x 개발자 안내서를 참조하세요.
- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - CreateBucket
  - PutBucketLifecycleConfiguration
  - ReceiveMessage
  - SendMessage

URI 구문 분석

다음 코드 예제에서는 버킷 이름 및 객체 키와 같은 중요한 구성 요소를 추출하기 위해 Amazon S3 URI를 구문 분석하는 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

S3Uri 클래스를 사용하여 Amazon S3 URI를 구문 분석합니다.

```
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);
        }
    }
```

S3 이벤트 알림 처리

다음 코드 예시에서는 객체 지향적 방식으로 S3 이벤트 알림을 사용하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예시에서는 Amazon SQS를 사용하여 S3 알림 이벤트를 처리하는 방법을 보여줍니다.

```
    /**
     * This method receives S3 event notifications by using an SqsAsyncClient.
     * After the client receives the messages it deserializes the JSON payload and
 logs them. It uses
     * the S3EventNotification class (part of the S3 event notification API for
 Java) to deserialize
     * the JSON payload and access the messages in an object-oriented way.
     *
     * @param queueUrl The URL of the AWS SQS queue that receives the S3 event
 notifications.
     * @see <a href="https://sdk.amazonaws.com/java/api/latest/software/amazon/
awssdk/eventnotifications/s3/model/package-summary.html">S3EventNotification API</
a>.
     * <p>
     * To use S3 event notification serialization/deserialization to objects, add
 the following
     * dependency to your Maven pom.xml file.
     * <dependency>
     * <groupId>software.amazon.awssdk</groupId>
     * <artifactId>s3-event-notifications</artifactId>
     * <version><LATEST></version>
     * </dependency>
     * <p>
     * The S3 event notification API became available with version 2.25.11 of the
 Java SDK.
     * <p>
     * This example shows the use of the API with AWS SQS, but it can be used to
 process S3 event notifications
     * in AWS SNS or AWS Lambda as well.
```

```
    * <p>
    * Note: The S3EventNotification class does not work with messages routed
through AWS EventBridge.
    */
   static void processS3Events(String bucketName, String queueUrl, String queueArn)
{
       try {
           // Configure the bucket to send Object Created and Object Tagging
notifications to an existing SQS queue.
           s3Client.putBucketNotificationConfiguration(b -> b
                   .notificationConfiguration(ncb -> ncb
                           .queueConfigurations(qcb -> qcb
                                   .events(Event.S3_OBJECT_CREATED,
Event.S3_OBJECT_TAGGING)
                                   .queueArn(queueArn)))
                           .bucket(bucketName)
           ).join();

           triggerS3EventNotifications(bucketName);
           // Wait for event notifications to propagate.
           Thread.sleep(Duration.ofSeconds(5).toMillis());

           boolean didReceiveMessages = true;
           while (didReceiveMessages) {
               // Display the number of messages that are available in the queue.
               sqsClient.getQueueAttributes(b -> b
                               .queueUrl(queueUrl)

.attributeNames(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)
                       ).thenAccept(attributeResponse ->
                               logger.info("Approximate number of messages in the
queue: {}",

attributeResponse.attributes().get(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)))
                       .join();

               // Receive the messages.
               ReceiveMessageResponse response = sqsClient.receiveMessage(b -> b
                       .queueUrl(queueUrl)
               ).get();
               logger.info("Count of received messages: {}",
response.messages().size());
               didReceiveMessages = !response.messages().isEmpty();
```

```
                // Create a collection to hold the received message for deletion
                // after we log the messages.
                HashSet<DeleteMessageBatchRequestEntry> messagesToDelete = new
HashSet<>();
                // Process each message.
                response.messages().forEach(message -> {
                    logger.info("Message id: {}", message.messageId());
                    // Deserialize JSON message body to a S3EventNotification object
                    // to access messages in an object-oriented way.
                    S3EventNotification event =
S3EventNotification.fromJson(message.body());

                    // Log the S3 event notification record details.
                    if (event.getRecords() != null) {
                        event.getRecords().forEach(record -> {
                            String eventName = record.getEventName();
                            String key = record.getS3().getObject().getKey();
                            logger.info(record.toString());
                            logger.info("Event name is {} and key is {}", eventName,
key);
                        });
                    }
                    // Add logged messages to collection for batch deletion.
                    messagesToDelete.add(DeleteMessageBatchRequestEntry.builder()
                            .id(message.messageId())
                            .receiptHandle(message.receiptHandle())
                            .build());
                });
                // Delete messages.
                if (!messagesToDelete.isEmpty()) {
                    sqsClient.deleteMessageBatch(DeleteMessageBatchRequest.builder()
                            .queueUrl(queueUrl)
                            .entries(messagesToDelete)
                            .build()
                    ).join();
                }
            } // End of while block.
        } catch (InterruptedException | ExecutionException e) {
            throw new RuntimeException(e);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - DeleteMessageBatch
  - GetQueueAttributes
  - PutBucketNotificationConfiguration
  - ReceiveMessage

EventBridge에 이벤트 알림 전송

다음 코드 예시에서는 S3 이벤트 알림을 EventBridge로 보내고 알림을 Amazon SNS 주제 및 Amazon SQS 대기열로 라우팅하도록 버킷을 설정하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /** This method configures a bucket to send events to AWS EventBridge and
creates a rule
    * to route the S3 object created events to a topic and a queue.
    *
    * @param bucketName Name of existing bucket
    * @param topicArn ARN of existing topic to receive S3 event notifications
    * @param queueArn ARN of existing queue to receive S3 event notifications
    *
    *  An AWS CloudFormation stack sets up the bucket, queue, topic before the
method runs.
    */
   public static String setBucketNotificationToEventBridge(String bucketName,
String topicArn, String queueArn) {
       try {
           // Enable bucket to emit S3 Event notifications to EventBridge.
           s3Client.putBucketNotificationConfiguration(b -> b
                   .bucket(bucketName)
                   .notificationConfiguration(b1 -> b1
                           .eventBridgeConfiguration(
                                   SdkBuilder::build)
                   ).build()).join();
```

```
            // Create an EventBridge rule to route Object Created notifications.
            PutRuleRequest putRuleRequest = PutRuleRequest.builder()
                    .name(RULE_NAME)
                    .eventPattern("""
                            {
                              "source": ["aws.s3"],
                              "detail-type": ["Object Created"],
                              "detail": {
                                "bucket": {
                                  "name": ["%s"]
                                }
                              }
                            }
                            """.formatted(bucketName))
                    .build();

            // Add the rule to the default event bus.
            PutRuleResponse putRuleResponse =
eventBridgeClient.putRule(putRuleRequest)
                    .whenComplete((r, t) -> {
                        if (t != null) {
                            logger.error("Error creating event bus rule: " +
t.getMessage(), t);
                            throw new RuntimeException(t.getCause().getMessage(),
t);
                        }
                        logger.info("Event bus rule creation request sent
successfully. ARN is: {}", r.ruleArn());
                    }).join();

            // Add the existing SNS topic and SQS queue as targets to the rule.
            eventBridgeClient.putTargets(b -> b
                    .eventBusName("default")
                    .rule(RULE_NAME)
                    .targets(List.of (
                            Target.builder()
                                    .arn(queueArn)
                                    .id("Queue")
                                    .build(),
                            Target.builder()
                                    .arn(topicArn)
                                    .id("Topic")
                                    .build())
```

```
                            )
                    ).join();
            return putRuleResponse.ruleArn();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - PutBucketNotificationConfiguration
  - PutRule
  - PutTargets

업로드 및 다운로드 추적

다음 코드 예시는 Amazon S3 객체 업로드 또는 다운로드를 추적하는 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

파일 업로드 진행 상황을 추적합니다.

```
    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!
        */
    }
```

파일 다운로드 진행 상황을 추적합니다.

```
    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!
        */
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - GetObject
  - PutObject

버킷에 디렉터리 업로드

다음 코드 예제에서는 Amazon Simple Storage Service(Amazon S3) 버킷에 로컬 디렉터리를 반복적으로 업로드하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

S3TransferManager를 사용하여 로컬 디렉터리를 업로드합니다. 파일 전체를 보고 테스트합니다.

```
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();
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 UploadDirectory를 참조하세요.

대용량 파일 업로드 또는 다운로드

다음 코드 예제는 Amazon S3에 대용량 파일을 업로드하고 Amazon S3에서 대용량 파일을 다운로드하는 방법을 보여줍니다.

자세한 내용은 멀티파트 업로드를 사용하여 객체 업로드를 참조하세요.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

S3TransferManager를 사용하여 S3 버킷과 파일을 주고받는 함수를 호출합니다.

```
    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();
```

```
    }
```

전체 로컬 디렉토리를 업로드합니다.

```
    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();
    }
```

단일 파일을 업로드합니다.

```
    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();
    }
```

코드 예제에서는 다음 가져오기를 사용합니다.

```
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;
```

콘텐츠 크기가 임계값을 초과할 때 AWS CRT 기반 S3 클라이언트 위에 있는 S3 Transfer Manager를 사용하여 멀티파트 업로드를 투명하게 수행할 수 있습니다. 기본 임계값 크기는 8MB입니다.

```
    /**
     * Uploads a file to an Amazon S3 bucket using the S3TransferManager.
     *
     * @param filePath the file path of the file to be uploaded
     */
    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();
    }
```

멀티파트 업로드를 수행하려면 S3Client API를 사용합니다.

```
    /**
     * Performs a multipart upload to Amazon S3 using the provided S3 client.
     *
     * @param filePath the path to the file to be uploaded
     */
    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()));
    }
```

멀티파트 지원이 활성화된 S3AsyncClient API를 사용하여 멀티파트 업로드를 수행합니다.

```
    /**
     * Uploads a file to an S3 bucket using the S3AsyncClient and enabling multipart
support.
     *
     * @param filePath the local file path of the file to be uploaded
     */
    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.");
    }
```

알 수 없는 크기의 스트림 업로드

다음 코드 예제는 알 수 없는 크기의 스트림을 Amazon S3 객체에 업로드하는 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

AWS CRT 기반 S3 클라이언트를 사용합니다.

```
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.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutorService;
```

```
import java.util.concurrent.Executors;

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

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket-" + UUID.randomUUID(); // Change
 bucket name.
        String key = UUID.randomUUID().toString();

        AsyncExampleUtils.createBucket(bucketName);
        try {
            PutObjectFromStreamAsync example = new PutObjectFromStreamAsync();
            S3AsyncClient s3AsyncClientCrt = S3AsyncClient.crtCreate();
            PutObjectResponse putObjectResponse =
 example.putObjectFromStreamCrt(s3AsyncClientCrt, bucketName, key);
            logger.info("Object {} etag: {}", key, putObjectResponse.eTag());
            logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
        } finally {
            AsyncExampleUtils.deleteObject(bucketName, key);
            AsyncExampleUtils.deleteBucket(bucketName);
        }
    }

    /**
     * @param s33CrtAsyncClient - To upload content from a stream of unknown size,
 use can the AWS CRT-based S3 client.
     * @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 putObjectFromStreamCrt(S3AsyncClient s33CrtAsyncClient,
 String bucketName, String key) {

        // AsyncExampleUtils.randomString() returns a random string up to 100
 characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
 randomString.length());
        InputStream inputStream = new ByteArrayInputStream(randomString.getBytes());
```

```
        // Executor required to handle reading from the InputStream on a separate
 thread so the main upload is not blocked.
        ExecutorService executor = Executors.newSingleThreadExecutor();
        // Specify `null` for the content length when you don't know the content
 length.
        AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream, null,
 executor);

        CompletableFuture<PutObjectResponse> responseFuture =
                s33CrtAsyncClient.putObject(r -> r.bucket(bucketName).key(key),
 body);

        PutObjectResponse response = responseFuture.join(); // Wait for the
 response.
        logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        executor.shutdown();
        return response;
    }
}
```

멀티파트 업로드가 활성화된 표준 비동기 S3 클라이언트를 사용합니다.

```
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.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

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

```
   public static void main(String[] args) {
       String bucketName = "amzn-s3-demo-bucket-" + UUID.randomUUID(); // Change
bucket name.
       String key = UUID.randomUUID().toString();

       AsyncExampleUtils.createBucket(bucketName);
       try {
           PutObjectFromStreamAsyncMp example = new PutObjectFromStreamAsyncMp();
           S3AsyncClient s3AsyncClientMp =
S3AsyncClient.builder().multipartEnabled(true).build();
           PutObjectResponse putObjectResponse =
example.putObjectFromStreamMp(s3AsyncClientMp, bucketName, key);
           logger.info("Object {} etag: {}", key, putObjectResponse.eTag());
           logger.info("Object {} uploaded to bucket {}.", key, bucketName);
       } catch (SdkException e) {
           logger.error(e.getMessage(), e);
       } finally {
           AsyncExampleUtils.deleteObject(bucketName, key);
           AsyncExampleUtils.deleteBucket(bucketName);
       }
   }

   /**
    * @param s3AsyncClientMp - To upload content from a stream of unknown size, use
can the S3 asynchronous client with multipart enabled.
    * @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 putObjectFromStreamMp(S3AsyncClient s3AsyncClientMp,
String bucketName, String key) {

       // AsyncExampleUtils.randomString() returns a random string up to 100
characters.
       String randomString = AsyncExampleUtils.randomString();
       logger.info("random string to upload: {}: length={}", randomString,
randomString.length());
       InputStream inputStream = new ByteArrayInputStream(randomString.getBytes());

       // Executor required to handle reading from the InputStream on a separate
thread so the main upload is not blocked.
       ExecutorService executor = Executors.newSingleThreadExecutor();
```

```
        // Specify `null` for the content length when you don't know the content
 length.
        AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream, null,
 executor);

        CompletableFuture<PutObjectResponse> responseFuture =
                s3AsyncClientMp.putObject(r -> r.bucket(bucketName).key(key), body);

        PutObjectResponse response = responseFuture.join(); // Wait for the
 response.
        logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        executor.shutdown();
        return response;
    }
}
```

Amazon S3 Transfer Manager를 사용합니다.

```
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.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.io.InputStream;
import java.util.UUID;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

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

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket" + UUID.randomUUID();
        String key = UUID.randomUUID().toString();
```

```
        AsyncExampleUtils.createBucket(bucketName);
        try {
            UploadStream example = new UploadStream();
            CompletedUpload completedUpload =
example.uploadStream(S3TransferManager.create(), bucketName, key);
            logger.info("Object {} etag: {}", key,
completedUpload.response().eTag());
            logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
        } finally {
            AsyncExampleUtils.deleteObject(bucketName, key);
            AsyncExampleUtils.deleteBucket(bucketName);
        }
    }

    /**
     * @param transferManager - To upload content from a stream of unknown size, you
can use the S3TransferManager based on the AWS CRT-based S3 client.
     * @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) {

        // AsyncExampleUtils.randomString() returns a random string up to 100
characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
randomString.length());
        InputStream inputStream = new ByteArrayInputStream(randomString.getBytes());

        // Executor required to handle reading from the InputStream on a separate
thread so the main upload is not blocked.
        ExecutorService executor = Executors.newSingleThreadExecutor();
        // Specify `null` for the content length when you don't know the content
length.
        AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream, null,
executor);

        Upload upload = transferManager.upload(builder -> builder
                .requestBody(body)
```

```
                .putObjectRequest(req -> req.bucket(bucketName).key(key))
                .build());

        CompletedUpload completedUpload = upload.completionFuture().join();
        executor.shutdown();
        return completedUpload;
    }
}
```

체크섬 사용

다음 코드 예제에서는 체크섬을 사용하여 Amazon S3 객체로 작업하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

코드 예제에서는 다음 가져오기 하위 집합을 사용합니다.

```
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;
```

PutObjectRequest를 빌드할 때 putObject 메서드에 대한 체크섬 알고리즘을 지정합니다.

```
    public void putObjectWithChecksum() {
        s3Client.putObject(b -> b
                .bucket(bucketName)
                .key(key)
                .checksumAlgorithm(ChecksumAlgorithm.CRC32),
            RequestBody.fromString("This is a test"));
    }
```

GetObjectRequest를 구축할 때 getObject 메서드의 체크섬을 확인합니다.

```
    public GetObjectResponse getObjectWithChecksum() {
        return s3Client.getObject(b -> b
                .bucket(bucketName)
                .key(key)
                .checksumMode(ChecksumMode.ENABLED))
            .response();
    }
```

PutObjectRequest를 빌드할 때 putObject 메서드에 대한 체크섬을 미리 계산합니다.

```
    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)));
    }
```

콘텐츠 크기가 임계값을 초과할 때 AWS CRT 기반 S3 클라이언트 위에 있는 S3 Transfer Manager를 사용하여 멀티파트 업로드를 투명하게 수행할 수 있습니다. 기본 임계값 크기는 8MB입니다.

SDK에서 사용할 체크섬 알고리즘을 지정할 수 있습니다. 기본적으로 SDK는 CRC32 알고리즘을 사용합니다.

```
    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();
    }
```

멀티파트 업로드를 수행하려면 S3Client API 또는 (S3AsyncClient API)를 사용합니다. 추가 체크섬을 지정하는 경우 업로드를 시작할 때 사용할 알고리즘을 지정해야 합니다. 또한 각 파트 요청에 대한 알고리즘을 지정하고 업로드 후 각 파트에 대해 계산된 체크섬을 제공해야 합니다.

```
    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()));
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 항목을 참조하세요.
  - CompleteMultipartUpload
  - CreateMultipartUpload
  - UploadPart

서버리스 예제

Amazon S3 트리거를 사용하여 Lambda 함수 간접 호출

다음 코드 예제는 S3 버킷에 객체를 업로드하여 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 해당 함수는 이벤트 파라미터에서 S3 버킷 이름과 객체 키를 검색하고 Amazon S3 API를 호출하여 객체의 콘텐츠 유형을 검색하고 로깅합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Java를 사용하여 Lambda로 S3 이벤트를 사용합니다.

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

SDK for Java 2.x를 사용한 Amazon S3 Control 예제

다음 코드 예제에서는 Amazon S3 Control과 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시작하기
- 기본 사항
- 작업

시작하기

Amazon S3 Control 시작

다음 코드 예제에서는 Amazon S3 Control 사용을 시작하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryMode;
import software.amazon.awssdk.core.retry.RetryPolicy;
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.s3control.S3ControlAsyncClient;
import software.amazon.awssdk.services.s3control.model.JobListDescriptor;
import software.amazon.awssdk.services.s3control.model.JobStatus;
import software.amazon.awssdk.services.s3control.model.ListJobsRequest;
import software.amazon.awssdk.services.s3control.paginators.ListJobsPublisher;
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this example:
 * <p/>
 * The SDK must be able to authenticate AWS requests on your behalf. If you have not
 configured
 * authentication for SDKs and tools,see https://docs.aws.amazon.com/sdkref/latest/
guide/access.html in the AWS SDKs and Tools Reference Guide.
 * <p/>
 * You must have a runtime environment configured with the Java SDK.
 * See https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html in
 the Developer Guide if this is not set up.
 */
public class HelloS3Batch {
    private static S3ControlAsyncClient asyncClient;

    public static void main(String[] args) {
        S3BatchActions actions = new S3BatchActions();
        String accountId = actions.getAccountId();
        try {
            listBatchJobsAsync(accountId)
                .exceptionally(ex -> {
                    System.err.println("List batch jobs failed: " +
 ex.getMessage());
                    return null;
                })
```

```
                .join();

        } catch (CompletionException ex) {
            System.err.println("Failed to list batch jobs: " + ex.getMessage());
        }
    }

    /**
     * Retrieves the asynchronous S3 Control client instance.
     * <p>
     * This method creates and returns a singleton instance of the {@link
 S3ControlAsyncClient}. If the instance
     * has not been created yet, it will be initialized with the following
 configuration:
     * <ul>
     *   <li>Maximum concurrency: 100</li>
     *   <li>Connection timeout: 60 seconds</li>
     *   <li>Read timeout: 60 seconds</li>
     *   <li>Write timeout: 60 seconds</li>
     *   <li>API call timeout: 2 minutes</li>
     *   <li>API call attempt timeout: 90 seconds</li>
     *   <li>Retry policy: 3 retries</li>
     *   <li>Region: US_EAST_1</li>
     *   <li>Credentials provider: {@link EnvironmentVariableCredentialsProvider}</
li>
     * </ul>
     *
     * @return the asynchronous S3 Control client instance
     */
    private static S3ControlAsyncClient getAsyncClient() {
        if (asyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();
```

```
            asyncClient = S3ControlAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return asyncClient;
    }

    /**
     * Asynchronously lists batch jobs that have completed for the specified
account.
     *
     * @param accountId the ID of the account to list jobs for
     * @return a CompletableFuture that completes when the job listing operation is
finished
     */
    public static CompletableFuture<Void> listBatchJobsAsync(String accountId) {
        ListJobsRequest jobsRequest = ListJobsRequest.builder()
            .jobStatuses(JobStatus.COMPLETE)
            .accountId(accountId)
            .maxResults(10)
            .build();

        ListJobsPublisher publisher =
getAsyncClient().listJobsPaginator(jobsRequest);
        return publisher.subscribe(response -> {
            List<JobListDescriptor> jobs = response.jobs();
            for (JobListDescriptor job : jobs) {
                System.out.println("The job id is " + job.jobId());
                System.out.println("The job priority is " + job.priority());
            }
        }).thenAccept(response -> {
            System.out.println("Listing batch jobs completed");
        }).exceptionally(ex -> {
            System.err.println("Failed to list batch jobs: " + ex.getMessage());
            throw new RuntimeException(ex);
        });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListJobs를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 Amazon S3 Control의 핵심 작업을 학습하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

핵심 작업에 대해 알아봅니다.

```
package com.example.s3.batch;

import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.IOException;
import java.util.Map;
import java.util.Scanner;
import java.util.UUID;
import java.util.concurrent.CompletionException;

public class S3BatchScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final String STACK_NAME = "MyS3Stack";
    public static void main(String[] args) throws IOException {
        S3BatchActions actions = new S3BatchActions();
        String accountId = actions.getAccountId();
        String uuid = java.util.UUID.randomUUID().toString();
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon S3 Batch basics scenario.");
        System.out.println("""
            S3 Batch operations enables efficient and cost-effective processing of
 large-scale
            data stored in Amazon S3. It automatically scales resources to handle
 varying workloads
            without the need for manual intervention.
```

```
            One of the key features of S3 Batch is its ability to perform tagging
 operations on objects stored in
            S3 buckets. Users can leverage S3 Batch to apply, update, or remove tags
 on thousands or millions of
            objects in a single operation, streamlining the management and
 organization of their data.

            This can be particularly useful for tasks such as cost allocation,
 lifecycle management, or
            metadata-driven workflows, where consistent and accurate tagging is
 essential.
            S3 Batch's scalability and serverless nature make it an ideal solution
 for organizations with
            growing data volumes and complex data management requirements.

            This Java program walks you through Amazon S3 Batch operations.

            Let's get started...

            """);
        waitForInputToContinue(scanner);
        // Use CloudFormation to stand up the resource required for this scenario.
        System.out.println("Use CloudFormation to stand up the resource required for
 this scenario.");
        CloudFormationHelper.deployCloudFormationStack(STACK_NAME);

        Map<String, String> stackOutputs =
 CloudFormationHelper.getStackOutputs(STACK_NAME);
        String iamRoleArn = stackOutputs.get("S3BatchRoleArn");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Setup the required bucket for this scenario.");
        waitForInputToContinue(scanner);
        String bucketName = "amzn-s3-demo-bucket-" + UUID.randomUUID(); // Change
 bucket name.
        actions.createBucket(bucketName);
        String reportBucketName = "arn:aws:s3:::"+bucketName;
        String manifestLocation = "arn:aws:s3:::"+bucketName+"/job-manifest.csv";
        System.out.println("Populate the bucket with the required files.");
        String[] fileNames = {"job-manifest.csv", "object-key-1.txt", "object-
key-2.txt", "object-key-3.txt", "object-key-4.txt"};
        actions.uploadFilesToBucket(bucketName, fileNames, actions);
```

```
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create a S3 Batch Job");
        System.out.println("This job tags all objects listed in the manifest file
with tags");
        waitForInputToContinue(scanner);
        String jobId ;
        try {
            jobId = actions.createS3JobAsync(accountId, iamRoleArn,
manifestLocation, reportBucketName, uuid).join();
            System.out.println("The Job id is " + jobId);

        } catch (S3Exception e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }

        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Update an existing S3 Batch Operations job's
priority");
        System.out.println("""
             In this step, we modify the job priority value. The higher the number,
the higher the priority.
             So, a job with a priority of `30` would have a higher priority than a
job with
             a priority of `20`. This is a common way to represent the priority of a
task
             or job, with higher numbers indicating a higher priority.

             Ensure that the job status allows for priority updates. Jobs in
certain
             states (e.g., Cancelled, Failed, or Completed) cannot have their
priorities
             updated. Only jobs in the Active or Suspended state typically allow
priority
             updates.
```

```
             """);

        try {
            actions.updateJobPriorityAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Update job priority failed: " +
ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to update job priority: " + ex.getMessage());
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Cancel the S3 Batch job");
        System.out.print("Do you want to cancel the Batch job? (y/n): ");
        String cancelAns = scanner.nextLine();
        if (cancelAns != null && cancelAns.trim().equalsIgnoreCase("y")) {
            try {
                actions.cancelJobAsync(jobId, accountId)
                    .exceptionally(ex -> {
                        System.err.println("Cancel job failed: " + ex.getMessage());
                        return null;
                    })
                    .join();
            } catch (CompletionException ex) {
                System.err.println("Failed to cancel job: " + ex.getMessage());
            }
        } else {
            System.out.println("Job " +jobId +" was not canceled.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Describe the job that was just created");
        waitForInputToContinue(scanner);
        try {
            actions.describeJobAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Describe job failed: " + ex.getMessage());
                    return null;
```

```
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to describe job: " + ex.getMessage());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Describe the tags associated with the job");
        waitForInputToContinue(scanner);
        try {
            actions.getJobTagsAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Get job tags failed: " + ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to get job tags: " + ex.getMessage());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Update Batch Job Tags");
        waitForInputToContinue(scanner);
        try {
            actions.putJobTaggingAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Put job tagging failed: " +
ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to put job tagging: " + ex.getMessage());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the Amazon S3 Batch job tagging.");
        System.out.print("Do you want to delete Batch job tagging? (y/n)");
        String delAns = scanner.nextLine();
        if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
            try {
```

```
                actions.deleteBatchJobTagsAsync(jobId, accountId)
                    .exceptionally(ex -> {
                        System.err.println("Delete batch job tags failed: " +
ex.getMessage());
                        return null;
                    })
                    .join();
            } catch (CompletionException ex) {
                System.err.println("Failed to delete batch job tags: " +
ex.getMessage());
            }
        } else {
            System.out.println("Tagging was not deleted.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.print("Do you want to delete the AWS resources used in this
scenario? (y/n)");
        String delResAns = scanner.nextLine();
        if (delResAns != null && delResAns.trim().equalsIgnoreCase("y")) {
            actions.deleteFilesFromBucket(bucketName, fileNames, actions);
            actions.deleteBucketFolderAsync(bucketName);
            actions.deleteBucket(bucketName)
                .thenRun(() -> System.out.println("Bucket deletion completed"))
                .exceptionally(ex -> {
                    System.err.println("Error occurred: " + ex.getMessage());
                    return null;
                });
            CloudFormationHelper.destroyCloudFormationStack(STACK_NAME);
        } else {
            System.out.println("The AWS resources were not deleted.");
        }
        System.out.println("The Amazon S3 Batch scenario has successfully
completed.");
        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.");
            }
        }
    }


}
```

작업을 래핑하는 작업 클래스를 만듭니다.

```
public class S3BatchActions {

    private static S3ControlAsyncClient asyncClient;

    private static S3AsyncClient s3AsyncClient ;
    /**
     * Retrieves the asynchronous S3 Control client instance.
     * <p>
     * This method creates and returns a singleton instance of the {@link
 S3ControlAsyncClient}. If the instance
     * has not been created yet, it will be initialized with the following
 configuration:
     * <ul>
     *   <li>Maximum concurrency: 100</li>
     *   <li>Connection timeout: 60 seconds</li>
     *   <li>Read timeout: 60 seconds</li>
     *   <li>Write timeout: 60 seconds</li>
     *   <li>API call timeout: 2 minutes</li>
     *   <li>API call attempt timeout: 90 seconds</li>
     *   <li>Retry policy: 3 retries</li>
     *   <li>Region: US_EAST_1</li>
     *   <li>Credentials provider: {@link EnvironmentVariableCredentialsProvider}</
li>
     * </ul>
     *
     * @return the asynchronous S3 Control client instance
```

```
     */
    private static S3ControlAsyncClient getAsyncClient() {
        if (asyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryPolicy(RetryPolicy.builder()
                    .numRetries(3)
                    .build())
                .build();

            asyncClient = S3ControlAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return asyncClient;
    }

    private static S3AsyncClient getS3AsyncClient() {
        if (asyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();
```

```
            s3AsyncClient = S3AsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return s3AsyncClient;
    }


    /**
     * Cancels a job asynchronously.
     *
     * @param jobId The ID of the job to be canceled.
     * @param accountId The ID of the account associated with the job.
     * @return A {@link CompletableFuture} that completes when the job status has
been updated to "CANCELLED".
     *         If an error occurs during the update, the returned future will
complete exceptionally.
     */
    public CompletableFuture<Void> cancelJobAsync(String jobId, String accountId) {
        UpdateJobStatusRequest updateJobStatusRequest =
UpdateJobStatusRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .requestedJobStatus(String.valueOf(JobStatus.CANCELLED))
            .build();

        return asyncClient.updateJobStatus(updateJobStatusRequest)
            .thenAccept(updateJobStatusResponse -> {
                System.out.println("Job status updated to: " +
updateJobStatusResponse.status());
            })
            .exceptionally(ex -> {
                System.err.println("Failed to cancel job: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }

    /**
     * Updates the priority of a job asynchronously.
     *
     * @param jobId     the ID of the job to update
     * @param accountId the ID of the account associated with the job
```

```
    * @return a {@link CompletableFuture} that represents the asynchronous
operation, which completes when the job priority has been updated or an error has
occurred
    */
   public CompletableFuture<Void> updateJobPriorityAsync(String jobId, String
accountId) {
       UpdateJobPriorityRequest priorityRequest =
UpdateJobPriorityRequest.builder()
           .accountId(accountId)
           .jobId(jobId)
           .priority(60)
           .build();

       CompletableFuture<Void> future = new CompletableFuture<>();
       getAsyncClient().updateJobPriority(priorityRequest)
           .thenAccept(response -> {
               System.out.println("The job priority was updated");
               future.complete(null); // Complete the CompletableFuture on
successful execution
           })
           .exceptionally(ex -> {
               System.err.println("Failed to update job priority: " +
ex.getMessage());
               future.completeExceptionally(ex); // Complete the CompletableFuture
exceptionally on error
               return null; // Return null to handle the exception
           });

       return future;
   }

   /**
    * Asynchronously retrieves the tags associated with a specific job in an AWS
account.
    *
    * @param jobId     the ID of the job for which to retrieve the tags
    * @param accountId the ID of the AWS account associated with the job
    * @return a {@link CompletableFuture} that completes when the job tags have
been retrieved, or with an exception if the operation fails
    * @throws RuntimeException if an error occurs while retrieving the job tags
    */
   public CompletableFuture<Void> getJobTagsAsync(String jobId, String accountId) {
       GetJobTaggingRequest request = GetJobTaggingRequest.builder()
           .jobId(jobId)
```

```
            .accountId(accountId)
            .build();

        return asyncClient.getJobTagging(request)
            .thenAccept(response -> {
                List<S3Tag> tags = response.tags();
                if (tags.isEmpty()) {
                    System.out.println("No tags found for job ID: " + jobId);
                } else {
                    for (S3Tag tag : tags) {
                        System.out.println("Tag key is: " + tag.key());
                        System.out.println("Tag value is: " + tag.value());
                    }
                }
            })
            .exceptionally(ex -> {
                System.err.println("Failed to get job tags: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }

    /**
     * Asynchronously deletes the tags associated with a specific batch job.
     *
     * @param jobId     The ID of the batch job whose tags should be deleted.
     * @param accountId The ID of the account associated with the batch job.
     * @return A CompletableFuture that completes when the job tags have been
successfully deleted, or an exception is thrown if the deletion fails.
     */
    public CompletableFuture<Void> deleteBatchJobTagsAsync(String jobId, String
accountId) {
        DeleteJobTaggingRequest jobTaggingRequest =
DeleteJobTaggingRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .build();

        return asyncClient.deleteJobTagging(jobTaggingRequest)
            .thenAccept(response -> {
                System.out.println("You have successfully deleted " + jobId + "
tagging.");
            })
            .exceptionally(ex -> {
                System.err.println("Failed to delete job tags: " + ex.getMessage());
```

```
                throw new RuntimeException(ex);
            });
    }

    /**
     * Asynchronously describes the specified job.
     *
     * @param jobId     the ID of the job to describe
     * @param accountId the ID of the AWS account associated with the job
     * @return a {@link CompletableFuture} that completes when the job description
is available
     * @throws RuntimeException if an error occurs while describing the job
     */
    public CompletableFuture<Void> describeJobAsync(String jobId, String accountId)
{
        DescribeJobRequest jobRequest = DescribeJobRequest.builder()
            .jobId(jobId)
            .accountId(accountId)
            .build();

        return getAsyncClient().describeJob(jobRequest)
            .thenAccept(response -> {
                System.out.println("Job ID: " + response.job().jobId());
                System.out.println("Description: " + response.job().description());
                System.out.println("Status: " + response.job().statusAsString());
                System.out.println("Role ARN: " + response.job().roleArn());
                System.out.println("Priority: " + response.job().priority());
                System.out.println("Progress Summary: " +
response.job().progressSummary());

                // Print out details about the job manifest.
                JobManifest manifest = response.job().manifest();
                System.out.println("Manifest Location: " +
manifest.location().objectArn());
                System.out.println("Manifest ETag: " + manifest.location().eTag());

                // Print out details about the job operation.
                JobOperation operation = response.job().operation();
                if (operation.s3PutObjectTagging() != null) {
                    System.out.println("Operation: S3 Put Object Tagging");
                    System.out.println("Tag Set: " +
operation.s3PutObjectTagging().tagSet());
                }
```

```
                // Print out details about the job report.
                JobReport report = response.job().report();
                System.out.println("Report Bucket: " + report.bucket());
                System.out.println("Report Prefix: " + report.prefix());
                System.out.println("Report Format: " + report.format());
                System.out.println("Report Enabled: " + report.enabled());
                System.out.println("Report Scope: " + report.reportScopeAsString());
            })
            .exceptionally(ex -> {
                System.err.println("Failed to describe job: " + ex.getMessage());
                throw new RuntimeException(ex);
            });
    }

    /**
     * Creates an asynchronous S3 job using the AWS Java SDK.
     *
     * @param accountId         the AWS account ID associated with the job
     * @param iamRoleArn        the ARN of the IAM role to be used for the job
     * @param manifestLocation  the location of the job manifest file in S3
     * @param reportBucketName  the name of the S3 bucket to store the job report
     * @param uuid              a unique identifier for the job
     * @return a CompletableFuture that represents the asynchronous creation of the
 S3 job.
     *         The CompletableFuture will return the job ID if the job is created
 successfully,
     *         or throw an exception if there is an error.
     */
    public CompletableFuture<String> createS3JobAsync(String accountId, String
 iamRoleArn,
                                                       String manifestLocation,
 String reportBucketName, String uuid) {

        String[] bucketName = new String[]{""};
        String[] parts = reportBucketName.split(":::");
        if (parts.length > 1) {
            bucketName[0] = parts[1];
        } else {
            System.out.println("The input string does not contain the expected
 format.");
        }

        return CompletableFuture.supplyAsync(() -> getETag(bucketName[0], "job-
manifest.csv"))
```

```
            .thenCompose(eTag -> {
                  ArrayList<S3Tag> tagSet = new ArrayList<>();
                S3Tag s3Tag = S3Tag.builder()
                    .key("keyOne")
                    .value("ValueOne")
                    .build();
                S3Tag s3Tag2 = S3Tag.builder()
                    .key("keyTwo")
                    .value("ValueTwo")
                    .build();
                tagSet.add(s3Tag);
                tagSet.add(s3Tag2);

                S3SetObjectTaggingOperation objectTaggingOperation =
S3SetObjectTaggingOperation.builder()
                    .tagSet(tagSet)
                    .build();

                JobOperation jobOperation = JobOperation.builder()
                    .s3PutObjectTagging(objectTaggingOperation)
                    .build();

                JobManifestLocation jobManifestLocation =
JobManifestLocation.builder()
                    .objectArn(manifestLocation)
                    .eTag(eTag)
                    .build();

                JobManifestSpec manifestSpec = JobManifestSpec.builder()
                    .fieldsWithStrings("Bucket", "Key")
                    .format("S3BatchOperations_CSV_20180820")
                    .build();

                JobManifest jobManifest = JobManifest.builder()
                    .spec(manifestSpec)
                    .location(jobManifestLocation)
                    .build();

                JobReport jobReport = JobReport.builder()
                    .bucket(reportBucketName)
                    .prefix("reports")
                    .format("Report_CSV_20180820")
                    .enabled(true)
                    .reportScope("AllTasks")
```

```
                    .build();

                CreateJobRequest jobRequest = CreateJobRequest.builder()
                    .accountId(accountId)
                    .description("Job created using the AWS Java SDK")
                    .manifest(jobManifest)
                    .operation(jobOperation)
                    .report(jobReport)
                    .priority(42)
                    .roleArn(iamRoleArn)
                    .clientRequestToken(uuid)
                    .confirmationRequired(false)
                    .build();

                // Create the job asynchronously.
                 return getAsyncClient().createJob(jobRequest)
                    .thenApply(CreateJobResponse::jobId);
                 })
                 .handle((jobId, ex) -> {
                    if (ex != null) {
                    Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                    if (cause instanceof S3ControlException) {
                        throw new CompletionException(cause);
                    } else {
                        throw new RuntimeException(cause);
                    }
                }
                return jobId;
            });
    }

    /**
     * Retrieves the ETag (Entity Tag) for an object stored in an Amazon S3 bucket.
     *
     * @param bucketName the name of the Amazon S3 bucket where the object is stored
     * @param key the key (file name) of the object in the Amazon S3 bucket
     * @return the ETag of the object
     */
    public String getETag(String bucketName, String key) {
        S3Client s3Client = S3Client.builder()
            .region(Region.US_EAST_1)
            .build();
```

```
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        HeadObjectResponse headObjectResponse =
s3Client.headObject(headObjectRequest);
        return headObjectResponse.eTag();
    }

    /**
     * Asynchronously adds tags to a job in the system.
     *
     * @param jobId     the ID of the job to add tags to
     * @param accountId the account ID associated with the job
     * @return a CompletableFuture that completes when the tagging operation is
finished
     */
    public CompletableFuture<Void> putJobTaggingAsync(String jobId, String
accountId) {
        S3Tag departmentTag = S3Tag.builder()
            .key("department")
            .value("Marketing")
            .build();

        S3Tag fiscalYearTag = S3Tag.builder()
            .key("FiscalYear")
            .value("2020")
            .build();

        PutJobTaggingRequest putJobTaggingRequest = PutJobTaggingRequest.builder()
            .jobId(jobId)
            .accountId(accountId)
            .tags(departmentTag, fiscalYearTag)
            .build();

        return asyncClient.putJobTagging(putJobTaggingRequest)
            .thenRun(() -> {
                System.out.println("Additional Tags were added to job " + jobId);
            })
            .exceptionally(ex -> {
                System.err.println("Failed to add tags to job: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
```

```
    }

    // Setup the S3 bucket required for this scenario.
    /**
     * Creates an Amazon S3 bucket with the specified name.
     *
     * @param bucketName the name of the S3 bucket to create
     * @throws S3Exception if there is an error creating the bucket
     */
    public void createBucket(String bucketName) {
        try {
            S3Client s3Client = S3Client.builder()
                .region(Region.US_EAST_1)
                .build();


            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);
        }
    }

    /**
     * Uploads a file to an Amazon S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to upload the file to
     * @param fileName the name of the file to be uploaded
     * @throws RuntimeException if an error occurs during the file upload
```

```
     */
    public void populateBucket(String bucketName, String fileName) {
        // Define the path to the directory.
        Path filePath = Paths.get("src/main/resources/batch/",
 fileName).toAbsolutePath();
        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(fileName)
            .build();

        CompletableFuture<PutObjectResponse> future =
 getS3AsyncClient().putObject(putOb, AsyncRequestBody.fromFile(filePath));
        future.whenComplete((result, ex) -> {
            if (ex != null) {
                System.err.println("Error uploading file: " + ex.getMessage());
            } else {
                System.out.println("Successfully placed " + fileName + " into bucket
 " + bucketName);
            }
        }).join();
    }


    // Update the bucketName in CSV.
    public void updateCSV(String newValue) {
        Path csvFilePath = Paths.get("src/main/resources/batch/job-
manifest.csv").toAbsolutePath();
        try {
            // Read all lines from the CSV file.
            List<String> lines = Files.readAllLines(csvFilePath);

            // Update the first value in each line.
            List<String> updatedLines = lines.stream()
                .map(line -> {
                    String[] parts = line.split(",");
                    parts[0] = newValue;
                    return String.join(",", parts);
                })
                .collect(Collectors.toList());

            // Write the updated lines back to the CSV file
            Files.write(csvFilePath, updatedLines);
            System.out.println("CSV file updated successfully.");
        } catch (Exception e) {
```

```
            e.printStackTrace();
        }
    }

    /**
     * Deletes an object from an Amazon S3 bucket asynchronously.
     *
     * @param bucketName The name of the S3 bucket where the object is stored.
     * @param objectName The name of the object to be deleted.
     * @return A {@link CompletableFuture} that completes when the object has been
deleted,
     *         or throws a {@link RuntimeException} if an error occurs during the
deletion.
     */
    public CompletableFuture<Void> deleteBucketObjects(String bucketName, String
objectName) {
        ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();
        toDelete.add(ObjectIdentifier.builder()
            .key(objectName)
            .build());

        DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
            .bucket(bucketName)
            .delete(Delete.builder()
                .objects(toDelete).build())
            .build();

        return getS3AsyncClient().deleteObjects(dor)
            .thenAccept(result -> {
                System.out.println("The object was deleted!");
            })
            .exceptionally(ex -> {
                throw new RuntimeException("Error deleting object: " +
ex.getMessage(), ex);
            });
    }

    /**
     * Deletes a folder and all its contents asynchronously from an Amazon S3
bucket.
     *
     * @param bucketName the name of the S3 bucket containing the folder to be
deleted
```

```
    * @return a {@link CompletableFuture} that completes when the folder and its
contents have been deleted
    * @throws RuntimeException if any error occurs during the deletion process
    */
   public void deleteBucketFolderAsync(String bucketName) {
       String folderName = "reports/";
       ListObjectsV2Request request = ListObjectsV2Request.builder()
           .bucket(bucketName)
           .prefix(folderName)
           .build();

       CompletableFuture<ListObjectsV2Response> listObjectsFuture =
getS3AsyncClient().listObjectsV2(request);
       listObjectsFuture.thenCompose(response -> {
           List<CompletableFuture<DeleteObjectResponse>> deleteFutures =
response.contents().stream()
               .map(obj -> {
                   DeleteObjectRequest deleteRequest =
DeleteObjectRequest.builder()
                       .bucket(bucketName)
                       .key(obj.key())
                       .build();
                   return getS3AsyncClient().deleteObject(deleteRequest)
                       .thenApply(deleteResponse -> {
                           System.out.println("Deleted object: " + obj.key());
                           return deleteResponse;
                       });
               })
               .collect(Collectors.toList());

           return CompletableFuture.allOf(deleteFutures.toArray(new
CompletableFuture[0]))
               .thenCompose(v -> {
                   // Delete the folder.
                   DeleteObjectRequest deleteRequest =
DeleteObjectRequest.builder()
                       .bucket(bucketName)
                       .key(folderName)
                       .build();
                   return getS3AsyncClient().deleteObject(deleteRequest)
                       .thenApply(deleteResponse -> {
                           System.out.println("Deleted folder: " + folderName);
                           return deleteResponse;
                       });
```

```
                });
        }).join();
    }

    /**
     * Deletes an Amazon S3 bucket.
     *
     * @param bucketName the name of the bucket to delete
     * @return a {@link CompletableFuture} that completes when the bucket has been
deleted, or exceptionally if there is an error
     * @throws RuntimeException if there is an error deleting the bucket
     */
    public CompletableFuture<Void> deleteBucket(String bucketName) {
        S3AsyncClient s3Client = getS3AsyncClient();
        return s3Client.deleteBucket(DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build())
            .thenAccept(deleteBucketResponse -> {
                System.out.println(bucketName + " was deleted");
            })
            .exceptionally(ex -> {
                // Handle the exception or rethrow it.
                throw new RuntimeException("Failed to delete bucket: " + bucketName,
ex);
            });
    }

    /**
     * Uploads a set of files to an Amazon S3 bucket.
     *
     * @param bucketName the name of the S3 bucket to upload the files to
     * @param fileNames an array of file names to be uploaded
     * @param actions an instance of {@link S3BatchActions} that provides the
implementation for the necessary S3 operations
     * @throws IOException if there's an error creating the text files or uploading
the files to the S3 bucket
     */
    public static void uploadFilesToBucket(String bucketName, String[] fileNames,
S3BatchActions actions) throws IOException {
        actions.updateCSV(bucketName);
        createTextFiles(fileNames);
        for (String fileName : fileNames) {
            actions.populateBucket(bucketName, fileName);
        }
```

```
        System.out.println("All files are placed in the S3 bucket " + bucketName);
    }

    /**
     * Deletes the specified files from the given S3 bucket.
     *
     * @param bucketName the name of the S3 bucket
     * @param fileNames an array of file names to be deleted from the bucket
     * @param actions the S3BatchActions instance to be used for the file deletion
     * @throws IOException if an I/O error occurs during the file deletion
     */
    public void deleteFilesFromBucket(String bucketName, String[] fileNames,
S3BatchActions actions) throws IOException {
        for (String fileName : fileNames) {
                    actions.deleteBucketObjects(bucketName, fileName)
                   .thenRun(() -> System.out.println("Object deletion completed"))
                   .exceptionally(ex -> {
                       System.err.println("Error occurred: " + ex.getMessage());
                       return null;
                   });
        }
        System.out.println("All files have been deleted from the bucket " +
bucketName);
    }

    public static void createTextFiles(String[] fileNames) {
        String currentDirectory = System.getProperty("user.dir");
        String directoryPath = currentDirectory + "\\src\\main\\resources\\batch";
        Path path = Paths.get(directoryPath);

        try {
            // Create the directory if it doesn't exist.
            if (Files.notExists(path)) {
                Files.createDirectories(path);
                System.out.println("Created directory: " + path.toString());
            } else {
                System.out.println("Directory already exists: " + path.toString());
            }

            for (String fileName : fileNames) {
                // Check if the file is a .txt file.
                if (fileName.endsWith(".txt")) {
                    // Define the path for the new file.
                    Path filePath = path.resolve(fileName);
```

```
                    System.out.println("Attempting to create file: " +
 filePath.toString());

                    // Create and write content to the new file.
                    Files.write(filePath, "This is a test".getBytes());

                    // Verify the file was created.
                    if (Files.exists(filePath)) {
                        System.out.println("Successfully created file: " +
 filePath.toString());
                    } else {
                        System.out.println("Failed to create file: " +
 filePath.toString());
                    }
                }
            }

        } catch (IOException e) {
            System.err.println("An error occurred: " + e.getMessage());
            e.printStackTrace();
        }
    }

    public String getAccountId() {
        StsClient stsClient = StsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        GetCallerIdentityResponse callerIdentityResponse =
 stsClient.getCallerIdentity();
        return callerIdentityResponse.account();
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - CreateJob
  - DeleteJobTagging
  - DescribeJob
  - GetJobTagging
  - ListJobs

- PutJobTagging
- UpdateJobPriority
- UpdateJobStatus

작업

`CreateJob`

다음 코드 예시는 `CreateJob`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

비동기 S3 작업을 만듭니다.

```
   /**
    * Creates an asynchronous S3 job using the AWS Java SDK.
    *
    * @param accountId         the AWS account ID associated with the job
    * @param iamRoleArn        the ARN of the IAM role to be used for the job
    * @param manifestLocation  the location of the job manifest file in S3
    * @param reportBucketName  the name of the S3 bucket to store the job report
    * @param uuid              a unique identifier for the job
    * @return a CompletableFuture that represents the asynchronous creation of the
S3 job.
    *         The CompletableFuture will return the job ID if the job is created
successfully,
    *         or throw an exception if there is an error.
    */
   public CompletableFuture<String> createS3JobAsync(String accountId, String
iamRoleArn,
                                                      String manifestLocation,
String reportBucketName, String uuid) {

       String[] bucketName = new String[]{""};
       String[] parts = reportBucketName.split(":::");
```

```
        if (parts.length > 1) {
            bucketName[0] = parts[1];
        } else {
            System.out.println("The input string does not contain the expected
 format.");
        }

        return CompletableFuture.supplyAsync(() -> getETag(bucketName[0], "job-
manifest.csv"))
            .thenCompose(eTag -> {
                  ArrayList<S3Tag> tagSet = new ArrayList<>();
                S3Tag s3Tag = S3Tag.builder()
                    .key("keyOne")
                    .value("ValueOne")
                    .build();
                S3Tag s3Tag2 = S3Tag.builder()
                    .key("keyTwo")
                    .value("ValueTwo")
                    .build();
                tagSet.add(s3Tag);
                tagSet.add(s3Tag2);

                S3SetObjectTaggingOperation objectTaggingOperation =
 S3SetObjectTaggingOperation.builder()
                    .tagSet(tagSet)
                    .build();

                JobOperation jobOperation = JobOperation.builder()
                    .s3PutObjectTagging(objectTaggingOperation)
                    .build();

                JobManifestLocation jobManifestLocation =
 JobManifestLocation.builder()
                    .objectArn(manifestLocation)
                    .eTag(eTag)
                    .build();

                JobManifestSpec manifestSpec = JobManifestSpec.builder()
                    .fieldsWithStrings("Bucket", "Key")
                    .format("S3BatchOperations_CSV_20180820")
                    .build();

                JobManifest jobManifest = JobManifest.builder()
                    .spec(manifestSpec)
```

```
                    .location(jobManifestLocation)
                    .build();

                JobReport jobReport = JobReport.builder()
                    .bucket(reportBucketName)
                    .prefix("reports")
                    .format("Report_CSV_20180820")
                    .enabled(true)
                    .reportScope("AllTasks")
                    .build();

                CreateJobRequest jobRequest = CreateJobRequest.builder()
                    .accountId(accountId)
                    .description("Job created using the AWS Java SDK")
                    .manifest(jobManifest)
                    .operation(jobOperation)
                    .report(jobReport)
                    .priority(42)
                    .roleArn(iamRoleArn)
                    .clientRequestToken(uuid)
                    .confirmationRequired(false)
                    .build();

                // Create the job asynchronously.
                 return getAsyncClient().createJob(jobRequest)
                    .thenApply(CreateJobResponse::jobId);
                 })
                 .handle((jobId, ex) -> {
                    if (ex != null) {
                    Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                    if (cause instanceof S3ControlException) {
                        throw new CompletionException(cause);
                    } else {
                        throw new RuntimeException(cause);
                    }
                }
                return jobId;
            });
    }
```

규정 준수 보존 작업을 만듭니다.

```
    /**
     * Creates a compliance retention job in Amazon S3 Control.
     * <p>
     * A compliance retention job in Amazon S3 Control is a feature that allows you
 to
     * set a retention period for objects stored in an S3 bucket.
     * This feature is particularly useful for organizations that need to comply
 with
     * regulatory requirements or internal policies that mandate the retention of
 data for
     * a specific duration.
     *
     * @param s3ControlClient The S3ControlClient instance to use for the API call.
     * @return The job ID of the created compliance retention job.
     */
    public static String createComplianceRetentionJob(final S3ControlClient
 s3ControlClient, String roleArn, String bucketName, String accountId) {
        final String manifestObjectArn = "arn:aws:s3:::amzn-s3-demo-manifest-bucket/
compliance-objects-manifest.csv";
        final String manifestObjectVersionId = "your-object-version-Id";

        Instant jan2025 = Instant.parse("2025-01-01T00:00:00Z");
        JobOperation jobOperation = JobOperation.builder()
            .s3PutObjectRetention(S3SetObjectRetentionOperation.builder()
                .retention(S3Retention.builder()
                    .mode(S3ObjectLockRetentionMode.COMPLIANCE)
                    .retainUntilDate(jan2025)
                    .build())
                .build())
            .build();

        JobManifestLocation manifestLocation = JobManifestLocation.builder()
            .objectArn(manifestObjectArn)
            .eTag(manifestObjectVersionId)
            .build();

        JobManifestSpec manifestSpec = JobManifestSpec.builder()
            .fieldsWithStrings("Bucket", "Key")
            .format("S3BatchOperations_CSV_20180820")
            .build();

        JobManifest manifestToPublicApi = JobManifest.builder()
```

```
            .location(manifestLocation)
            .spec(manifestSpec)
            .build();

        // Report details.
        final String jobReportBucketArn = "arn:aws:s3:::" + bucketName;
        final String jobReportPrefix = "reports/compliance-objects-bops";

        JobReport jobReport = JobReport.builder()
            .enabled(true)
            .reportScope(JobReportScope.ALL_TASKS)
            .bucket(jobReportBucketArn)
            .prefix(jobReportPrefix)
            .format(JobReportFormat.REPORT_CSV_20180820)
            .build();

        final Boolean requiresConfirmation = true;
        final int priority = 10;
        CreateJobRequest request = CreateJobRequest.builder()
            .accountId(accountId)
            .description("Set compliance retain-until to 1 Jan 2025")
            .manifest(manifestToPublicApi)
            .operation(jobOperation)
            .priority(priority)
            .roleArn(roleArn)
            .report(jobReport)
            .confirmationRequired(requiresConfirmation)
            .build();

        // Create the job and get the result.
        CreateJobResponse result = s3ControlClient.createJob(request);
        return result.jobId();
    }
```

법적 보존 작업을 만듭니다.

```
    /**
     * Creates a compliance retention job in Amazon S3 Control.
     * <p>
     * A compliance retention job in Amazon S3 Control is a feature that allows you
to
```

```
     * set a retention period for objects stored in an S3 bucket.
     * This feature is particularly useful for organizations that need to comply
 with
     * regulatory requirements or internal policies that mandate the retention of
 data for
     * a specific duration.
     *
     * @param s3ControlClient The S3ControlClient instance to use for the API call.
     * @return The job ID of the created compliance retention job.
     */
    public static String createComplianceRetentionJob(final S3ControlClient
 s3ControlClient, String roleArn, String bucketName, String accountId) {
        final String manifestObjectArn = "arn:aws:s3:::amzn-s3-demo-manifest-bucket/
compliance-objects-manifest.csv";
        final String manifestObjectVersionId = "your-object-version-Id";

        Instant jan2025 = Instant.parse("2025-01-01T00:00:00Z");
        JobOperation jobOperation = JobOperation.builder()
            .s3PutObjectRetention(S3SetObjectRetentionOperation.builder()
                .retention(S3Retention.builder()
                    .mode(S3ObjectLockRetentionMode.COMPLIANCE)
                    .retainUntilDate(jan2025)
                    .build())
                .build())
            .build();

        JobManifestLocation manifestLocation = JobManifestLocation.builder()
            .objectArn(manifestObjectArn)
            .eTag(manifestObjectVersionId)
            .build();

        JobManifestSpec manifestSpec = JobManifestSpec.builder()
            .fieldsWithStrings("Bucket", "Key")
            .format("S3BatchOperations_CSV_20180820")
            .build();

        JobManifest manifestToPublicApi = JobManifest.builder()
            .location(manifestLocation)
            .spec(manifestSpec)
            .build();

        // Report details.
        final String jobReportBucketArn = "arn:aws:s3:::" + bucketName;
        final String jobReportPrefix = "reports/compliance-objects-bops";
```

```
        JobReport jobReport = JobReport.builder()
            .enabled(true)
            .reportScope(JobReportScope.ALL_TASKS)
            .bucket(jobReportBucketArn)
            .prefix(jobReportPrefix)
            .format(JobReportFormat.REPORT_CSV_20180820)
            .build();

        final Boolean requiresConfirmation = true;
        final int priority = 10;
        CreateJobRequest request = CreateJobRequest.builder()
            .accountId(accountId)
            .description("Set compliance retain-until to 1 Jan 2025")
            .manifest(manifestToPublicApi)
            .operation(jobOperation)
            .priority(priority)
            .roleArn(roleArn)
            .report(jobReport)
            .confirmationRequired(requiresConfirmation)
            .build();

        // Create the job and get the result.
        CreateJobResponse result = s3ControlClient.createJob(request);
        return result.jobId();
    }
```

새 거버넌스 보존 작업을 만듭니다.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateGovernanceRetentionJob {

    public static void main(String[]args) throws ParseException {
        final String usage = """
```

```
            Usage:
                <manifestObjectArn> <jobReportBucketArn> <roleArn> <accountId>
<manifestObjectVersionId>

            Where:
                manifestObjectArn - The Amazon Resource Name (ARN) of the S3 object
that contains the manifest file for the governance objects.\s
                bucketName - The ARN of the S3 bucket where the job report will be
stored.
                roleArn - The ARN of the IAM role that will be used to perform the
governance retention operation.
                accountId - Your AWS account Id.
                manifestObjectVersionId =  A unique value that is used as the `eTag`
property of the `JobManifestLocation` object.
            """;

        if (args.length != 4) {
            System.out.println(usage);
            return;
        }

        String manifestObjectArn = args[0];
        String jobReportBucketArn = args[1];
        String roleArn = args[2];
        String accountId = args[3];
        String manifestObjectVersionId = args[4];

        S3ControlClient s3ControlClient = S3ControlClient.create();
        createGovernanceRetentionJob(s3ControlClient, manifestObjectArn,
jobReportBucketArn, roleArn, accountId, manifestObjectVersionId);
    }

    public static String createGovernanceRetentionJob(final S3ControlClient
s3ControlClient, String manifestObjectArn, String jobReportBucketArn, String
roleArn, String accountId, String manifestObjectVersionId) throws ParseException {
        final JobManifestLocation manifestLocation = JobManifestLocation.builder()
            .objectArn(manifestObjectArn)
            .eTag(manifestObjectVersionId)
            .build();

        final JobManifestSpec manifestSpec = JobManifestSpec.builder()
            .format(JobManifestFormat.S3_BATCH_OPERATIONS_CSV_20180820)
            .fields(Arrays.asList(JobManifestFieldName.BUCKET,
JobManifestFieldName.KEY))
```

```
            .build();

        final JobManifest manifestToPublicApi = JobManifest.builder()
            .location(manifestLocation)
            .spec(manifestSpec)
            .build();

        final String jobReportPrefix = "reports/governance-objects";
        final JobReport jobReport = JobReport.builder()
            .enabled(true)
            .reportScope(JobReportScope.ALL_TASKS)
            .bucket(jobReportBucketArn)
            .prefix(jobReportPrefix)
            .format(JobReportFormat.REPORT_CSV_20180820)
            .build();

        final SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
        final Date jan30th = format.parse("30/01/2025");

        final S3SetObjectRetentionOperation s3SetObjectRetentionOperation =
S3SetObjectRetentionOperation.builder()
            .retention(S3Retention.builder()
                .mode(S3ObjectLockRetentionMode.GOVERNANCE)
                .retainUntilDate(jan30th.toInstant())
                .build())
            .build();

        final JobOperation jobOperation = JobOperation.builder()
            .s3PutObjectRetention(s3SetObjectRetentionOperation)
            .build();

        final Boolean requiresConfirmation = true;
        final int priority = 10;

        final CreateJobRequest request = CreateJobRequest.builder()
            .accountId(accountId)
            .description("Put governance retention")
            .manifest(manifestToPublicApi)
            .operation(jobOperation)
            .priority(priority)
            .roleArn(roleArn)
            .report(jobReport)
            .confirmationRequired(requiresConfirmation)
            .build();
```

```
        final CreateJobResponse result = s3ControlClient.createJob(request);
        return result.jobId();
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateJob을 참조하세요.

`DeleteJobTagging`

다음 코드 예시는 `DeleteJobTagging`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Asynchronously deletes the tags associated with a specific batch job.
     *
     * @param jobId     The ID of the batch job whose tags should be deleted.
     * @param accountId The ID of the account associated with the batch job.
     * @return A CompletableFuture that completes when the job tags have been
successfully deleted, or an exception is thrown if the deletion fails.
     */
    public CompletableFuture<Void> deleteBatchJobTagsAsync(String jobId, String
accountId) {
        DeleteJobTaggingRequest jobTaggingRequest =
DeleteJobTaggingRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .build();

        return asyncClient.deleteJobTagging(jobTaggingRequest)
            .thenAccept(response -> {
                System.out.println("You have successfully deleted " + jobId + "
tagging.");
```

```
            })
            .exceptionally(ex -> {
                System.err.println("Failed to delete job tags: " + ex.getMessage());
                throw new RuntimeException(ex);
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteJobTagging을 참조하세요.

`DescribeJob`

다음 코드 예시는 `DescribeJob`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Asynchronously describes the specified job.
     *
     * @param jobId     the ID of the job to describe
     * @param accountId the ID of the AWS account associated with the job
     * @return a {@link CompletableFuture} that completes when the job description
is available
     * @throws RuntimeException if an error occurs while describing the job
     */
    public CompletableFuture<Void> describeJobAsync(String jobId, String accountId)
{
        DescribeJobRequest jobRequest = DescribeJobRequest.builder()
            .jobId(jobId)
            .accountId(accountId)
            .build();

        return getAsyncClient().describeJob(jobRequest)
            .thenAccept(response -> {
                System.out.println("Job ID: " + response.job().jobId());
```

```
                System.out.println("Description: " + response.job().description());
                System.out.println("Status: " + response.job().statusAsString());
                System.out.println("Role ARN: " + response.job().roleArn());
                System.out.println("Priority: " + response.job().priority());
                System.out.println("Progress Summary: " +
response.job().progressSummary());

                // Print out details about the job manifest.
                JobManifest manifest = response.job().manifest();
                System.out.println("Manifest Location: " +
manifest.location().objectArn());
                System.out.println("Manifest ETag: " + manifest.location().eTag());

                // Print out details about the job operation.
                JobOperation operation = response.job().operation();
                if (operation.s3PutObjectTagging() != null) {
                    System.out.println("Operation: S3 Put Object Tagging");
                    System.out.println("Tag Set: " +
operation.s3PutObjectTagging().tagSet());
                }

                // Print out details about the job report.
                JobReport report = response.job().report();
                System.out.println("Report Bucket: " + report.bucket());
                System.out.println("Report Prefix: " + report.prefix());
                System.out.println("Report Format: " + report.format());
                System.out.println("Report Enabled: " + report.enabled());
                System.out.println("Report Scope: " + report.reportScopeAsString());
            })
            .exceptionally(ex -> {
                System.err.println("Failed to describe job: " + ex.getMessage());
                throw new RuntimeException(ex);
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeJob을 참조하세요.

`GetJobTagging`

다음 코드 예시는 `GetJobTagging`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Asynchronously retrieves the tags associated with a specific job in an AWS
account.
    *
    * @param jobId     the ID of the job for which to retrieve the tags
    * @param accountId the ID of the AWS account associated with the job
    * @return a {@link CompletableFuture} that completes when the job tags have
been retrieved, or with an exception if the operation fails
    * @throws RuntimeException if an error occurs while retrieving the job tags
    */
   public CompletableFuture<Void> getJobTagsAsync(String jobId, String accountId) {
       GetJobTaggingRequest request = GetJobTaggingRequest.builder()
           .jobId(jobId)
           .accountId(accountId)
           .build();

       return asyncClient.getJobTagging(request)
           .thenAccept(response -> {
               List<S3Tag> tags = response.tags();
               if (tags.isEmpty()) {
                   System.out.println("No tags found for job ID: " + jobId);
               } else {
                   for (S3Tag tag : tags) {
                       System.out.println("Tag key is: " + tag.key());
                       System.out.println("Tag value is: " + tag.value());
                   }
               }
           })
           .exceptionally(ex -> {
               System.err.println("Failed to get job tags: " + ex.getMessage());
               throw new RuntimeException(ex); // Propagate the exception
           });
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetJobTagging을 참조하세요.

`PutJobTagging`

다음 코드 예시는 `PutJobTagging`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Asynchronously adds tags to a job in the system.
     *
     * @param jobId     the ID of the job to add tags to
     * @param accountId the account ID associated with the job
     * @return a CompletableFuture that completes when the tagging operation is
finished
     */
    public CompletableFuture<Void> putJobTaggingAsync(String jobId, String
accountId) {
        S3Tag departmentTag = S3Tag.builder()
            .key("department")
            .value("Marketing")
            .build();

        S3Tag fiscalYearTag = S3Tag.builder()
            .key("FiscalYear")
            .value("2020")
            .build();

        PutJobTaggingRequest putJobTaggingRequest = PutJobTaggingRequest.builder()
            .jobId(jobId)
            .accountId(accountId)
            .tags(departmentTag, fiscalYearTag)
            .build();

        return asyncClient.putJobTagging(putJobTaggingRequest)
            .thenRun(() -> {
```

```
                System.out.println("Additional Tags were added to job " + jobId);
            })
            .exceptionally(ex -> {
                System.err.println("Failed to add tags to job: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutJobTagging을 참조하세요.

`UpdateJobPriority`

다음 코드 예시는 `UpdateJobPriority`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Updates the priority of a job asynchronously.
     *
     * @param jobId     the ID of the job to update
     * @param accountId the ID of the account associated with the job
     * @return a {@link CompletableFuture} that represents the asynchronous
operation, which completes when the job priority has been updated or an error has
occurred
     */
    public CompletableFuture<Void> updateJobPriorityAsync(String jobId, String
accountId) {
        UpdateJobPriorityRequest priorityRequest =
UpdateJobPriorityRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .priority(60)
            .build();

        CompletableFuture<Void> future = new CompletableFuture<>();
```

```
        getAsyncClient().updateJobPriority(priorityRequest)
            .thenAccept(response -> {
                System.out.println("The job priority was updated");
                future.complete(null); // Complete the CompletableFuture on
successful execution
            })
            .exceptionally(ex -> {
                System.err.println("Failed to update job priority: " +
ex.getMessage());
                future.completeExceptionally(ex); // Complete the CompletableFuture
exceptionally on error
                return null; // Return null to handle the exception
            });

        return future;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateJobPriority를 참조하세요.

`UpdateJobStatus`

다음 코드 예시는 `UpdateJobStatus`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Cancels a job asynchronously.
     *
     * @param jobId The ID of the job to be canceled.
     * @param accountId The ID of the account associated with the job.
     * @return A {@link CompletableFuture} that completes when the job status has
been updated to "CANCELLED".
     *         If an error occurs during the update, the returned future will
complete exceptionally.
     */
```

```
    public CompletableFuture<Void> cancelJobAsync(String jobId, String accountId) {
        UpdateJobStatusRequest updateJobStatusRequest =
UpdateJobStatusRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .requestedJobStatus(String.valueOf(JobStatus.CANCELLED))
            .build();

        return asyncClient.updateJobStatus(updateJobStatusRequest)
            .thenAccept(updateJobStatusResponse -> {
                System.out.println("Job status updated to: " +
updateJobStatusResponse.status());
            })
            .exceptionally(ex -> {
                System.err.println("Failed to cancel job: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateJobStatus를 참조하세요.

SDK for Java 2.x를 사용한 S3 디렉터리 버킷 예제

다음 코드 예제에서는 S3 디렉터리 버킷과 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시작하기
- 기본 사항

- 작업
- 시나리오

시작하기

Amazon S3 디렉터리 버킷 시작

다음 코드 예제에서는 Amazon S3 디렉터리 버킷 사용을 시작하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
package com.example.s3.directorybucket;


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.model.Bucket;
import software.amazon.awssdk.services.s3.model.BucketInfo;
import software.amazon.awssdk.services.s3.model.BucketType;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateBucketResponse;
import software.amazon.awssdk.services.s3.model.DataRedundancy;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsResponse;
import software.amazon.awssdk.services.s3.model.LocationInfo;
import software.amazon.awssdk.services.s3.model.LocationType;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.List;
import java.util.stream.Collectors;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;

/**
 * Before running this example:
 * <p>
 * The SDK must be able to authenticate AWS requests on your behalf. If you have
 * not configured
 * authentication for SDKs and tools, see
 * https://docs.aws.amazon.com/sdkref/latest/guide/access.html in the AWS SDKs
 * and Tools Reference Guide.
 * <p>
 * You must have a runtime environment configured with the Java SDK.
 * See
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html in
 * the Developer Guide if this is not set up.
 * <p>
 * To use S3 directory buckets, configure a gateway VPC endpoint. This is the
 * recommended method to enable directory bucket traffic without
 * requiring an internet gateway or NAT device. For more information on
 * configuring VPC gateway endpoints, visit
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-express-
networking.html#s3-express-networking-vpc-gateway.
 * <p>
 * Directory buckets are available in specific AWS Regions and Zones. For
 * details on Regions and Zones supporting directory buckets, see
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-express-
networking.html#s3-express-endpoints.
 */

public class HelloS3DirectoryBuckets {
    private static final Logger logger =
 LoggerFactory.getLogger(HelloS3DirectoryBuckets.class);

    public static void main(String[] args) {
        String bucketName = "test-bucket-" + System.currentTimeMillis() + "--usw2-
az1--x-s3";
        Region region = Region.US_WEST_2;
        String zone = "usw2-az1";
        S3Client s3Client = createS3Client(region);

        try {
            // Create the directory bucket
            createDirectoryBucket(s3Client, bucketName, zone);
```

```
            logger.info("Created bucket: {}", bucketName);

            // List all directory buckets
            List<String> bucketNames = listDirectoryBuckets(s3Client);
            bucketNames.forEach(name -> logger.info("Bucket Name: {}", name));
        } catch (S3Exception e) {
            logger.error("An error occurred during S3 operations: {} - Error code:
{}",
                    e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
        } finally {
            try {
                // Delete the created bucket
                deleteDirectoryBucket(s3Client, bucketName);
                logger.info("Deleted bucket: {}", bucketName);
            } catch (S3Exception e) {
                logger.error("Failed to delete the bucket due to S3 error: {} -
Error code: {}",
                        e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
            } catch (RuntimeException e) {
                logger.error("Failed to delete the bucket due to unexpected error:
{}", e.getMessage(), e);
            } finally {
                s3Client.close();
            }
        }
    }

    /**
     * Creates a new S3 directory bucket in a specified Zone (For example, a
     * specified Availability Zone in this code example).
     *
     * @param s3Client   The S3 client used to create the bucket
     * @param bucketName The name of the bucket to be created
     * @param zone       The region where the bucket will be created
     * @throws S3Exception if there's an error creating the bucket
     */
    public static void createDirectoryBucket(S3Client s3Client, String bucketName,
String zone) throws S3Exception {
        logger.info("Creating bucket: {}", bucketName);

        CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
```

```
                .location(LocationInfo.builder()
                        .type(LocationType.AVAILABILITY_ZONE)
                        .name(zone).build())
                .bucket(BucketInfo.builder()
                        .type(BucketType.DIRECTORY)
                        .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                        .build())
                .build();
        try {
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .createBucketConfiguration(bucketConfiguration).build();
            CreateBucketResponse response = s3Client.createBucket(bucketRequest);
            logger.info("Bucket created successfully with location: {}",
response.location());
        } catch (S3Exception e) {
            logger.error("Error creating bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }

    /**
     * Lists all S3 directory buckets.
     *
     * @param s3Client The S3 client used to interact with S3
     * @return A list of bucket names
     */
    public static List<String> listDirectoryBuckets(S3Client s3Client) {
        logger.info("Listing all directory buckets");

        try {
            // Create a ListBucketsRequest
            ListDirectoryBucketsRequest listBucketsRequest =
ListDirectoryBucketsRequest.builder().build();

            // Retrieve the list of buckets
            ListDirectoryBucketsResponse response =
s3Client.listDirectoryBuckets(listBucketsRequest);

            // Extract bucket names
            List<String> bucketNames = response.buckets().stream()
                    .map(Bucket::name)
```

```
                    .collect(Collectors.toList());

            return bucketNames;
        } catch (S3Exception e) {
            logger.error("Failed to list buckets: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }

    /**
     * Deletes the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the bucket to delete
     */
    public static void deleteDirectoryBucket(S3Client s3Client, String bucketName) {
        try {
            DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                    .bucket(bucketName)
                    .build();
            s3Client.deleteBucket(deleteBucketRequest);
        } catch (S3Exception e) {
            logger.error("Failed to delete bucket: " + bucketName + " - Error code:
 " + e.awsErrorDetails().errorCode(),
                    e);
            throw e;
        }
    }

}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - CreateBucket
  - ListDirectoryBuckets

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- VPC 및 VPC 엔드포인트를 설정합니다.
- S3 디렉터리 버킷 및 S3 Express One Zone 스토리지 클래스로 작업하도록 정책, 역할 및 사용자를 설정합니다.
- 두 개의 S3 클라이언트를 만듭니다.
- 두 개의 버킷 만들기
- 객체를 만들고 복사합니다.
- 성능 차이를 보여줍니다.
- 버킷을 채워 사전식 순서 차이를 표시합니다.
- 사용자에게 리소스를 정리할지 묻는 프롬프트를 표시합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon S3 기능을 보여주는 대화형 시나리오를 실행합니다.

```
public class S3DirectoriesScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    private static final Logger logger =
 LoggerFactory.getLogger(S3DirectoriesScenario.class);
    static Scanner scanner = new Scanner(System.in);

    private static S3AsyncClient mS3RegularClient;
    private static S3AsyncClient mS3ExpressClient;

    private static String mdirectoryBucketName;
    private static String mregularBucketName;
```

```
   private static String stackName = "cfn-stack-s3-express-basics--" +
UUID.randomUUID();

   private static String regularUser = "";
   private static String vpcId = "";
   private static String expressUser = "";

   private static String vpcEndpointId = "";

   private static final S3DirectoriesActions s3DirectoriesActions = new
S3DirectoriesActions();

   public static void main(String[] args) {
       try {
           s3ExpressScenario();
       } catch (RuntimeException e) {
           logger.info(e.getMessage());
       }
   }

   // Runs the scenario.
   private static void s3ExpressScenario() {
       logger.info(DASHES);
       logger.info("Welcome to the Amazon S3 Express Basics demo using AWS SDK for
Java V2.");
       logger.info("""
           Let's get started! First, please note that S3 Express One Zone works
best when working within the AWS infrastructure,
           specifically when working in the same Availability Zone (AZ). To see the
best results in this example and when you implement
           directory buckets into your infrastructure, it is best to put your
compute resources in the same AZ as your directory
           bucket.
           """);
       waitForInputToContinue(scanner);
       logger.info(DASHES);

       // Create an optional VPC and create 2 IAM users.
       UserNames userNames = createVpcUsers();
       String expressUserName = userNames.getExpressUserName();
       String regularUserName = userNames.getRegularUserName();

       //  Set up two S3 clients, one regular and one express,
```

```
        //  and two buckets, one regular and one directory.
        setupClientsAndBuckets(expressUserName, regularUserName);

        // Create an S3 session for the express S3 client and add objects to the
buckets.
        logger.info("Now let's add some objects to our buckets and demonstrate how
to work with S3 Sessions.");
        waitForInputToContinue(scanner);
        String bucketObject = createSessionAddObjects();

        // Demonstrate performance differences between regular and directory
buckets.
        demonstratePerformance(bucketObject);

        // Populate the buckets to show the lexicographical difference between
        // regular and express buckets.
        showLexicographicalDifferences(bucketObject);

        logger.info(DASHES);
        logger.info("That's it for our tour of the basic operations for S3 Express
One Zone.");
        logger.info("Would you like to cleanUp the AWS resources? (y/n): ");
        String response = scanner.next().trim().toLowerCase();
        if (response.equals("y")) {
            cleanUp(stackName);
        }
    }

    /*
      Delete resources created by this scenario.
    */
    public static void cleanUp(String stackName) {
        try {
            if (mdirectoryBucketName != null) {
                s3DirectoriesActions.deleteBucketAndObjectsAsync(mS3ExpressClient,
mdirectoryBucketName).join();
            }
            logger.info("Deleted directory bucket " + mdirectoryBucketName);
            mdirectoryBucketName = null;
            if (mregularBucketName != null) {
                s3DirectoriesActions.deleteBucketAndObjectsAsync(mS3RegularClient,
mregularBucketName).join();
            }
        } catch (CompletionException ce) {
```

```
            Throwable cause = ce.getCause();
            if (cause instanceof S3Exception) {
                logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
        }

        logger.info("Deleted regular bucket " + mregularBucketName);
        mregularBucketName = null;
        CloudFormationHelper.destroyCloudFormationStack(stackName);
    }

    private static void showLexicographicalDifferences(String bucketObject) {
        logger.info(DASHES);
        logger.info("""
            7. Populate the buckets to show the lexicographical (alphabetical)
difference
            when object names are listed. Now let's explore how directory buckets
store
            objects in a different manner to regular buckets. The key is in the
name
            "Directory". Where regular buckets store their key/value pairs in a
            flat manner, directory buckets use actual directories/folders.
            This allows for more rapid indexing, traversing, and therefore
            retrieval times!

            The more segmented your bucket is, with lots of
            directories, sub-directories, and objects, the more efficient it
becomes.
            This structural difference also causes `ListObject` operations to
behave
            differently, which can cause unexpected results. Let's add a few more
            objects in sub-directories to see how the output of
            ListObjects changes.
            """);

        waitForInputToContinue(scanner);

        //  Populate a few more files in each bucket so that we can use
        //  ListObjects and show the difference.
        String otherObject = "other/" + bucketObject;
        String altObject = "alt/" + bucketObject;
```

```
        String otherAltObject = "other/alt/" + bucketObject;

        try {
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, otherObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, otherObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, altObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, altObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, otherAltObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, otherAltObject, "").join();

        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof NoSuchBucketException) {
                logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
            return;
        }

        try {
            // List objects in both S3 buckets.
            List<String> dirBucketObjects =
s3DirectoriesActions.listObjectsAsync(mS3ExpressClient,
mdirectoryBucketName).join();
            List<String> regBucketObjects =
s3DirectoriesActions.listObjectsAsync(mS3RegularClient, mregularBucketName).join();

            logger.info("Directory bucket content");
            for (String obj : dirBucketObjects) {
                logger.info(obj);
            }

            logger.info("Regular bucket content");
            for (String obj : regBucketObjects) {
                logger.info(obj);
            }
```

```
        } catch (CompletionException e) {
            logger.error("Async operation failed: {} ", e.getCause().getMessage());
            return;
        }

        logger.info("""
            Notice how the regular bucket lists objects in lexicographical order,
while the directory bucket does not. This is
            because the regular bucket considers the whole "key" to be the object
identifier, while the directory bucket actually
            creates directories and uses the object "key" as a path to the object.
            """);
        waitForInputToContinue(scanner);
    }

    /**
     * Demonstrates the performance difference between downloading an object from a
directory bucket and a regular bucket.
     *
     * <p>This method:
     * <ul>
     *     <li>Prompts the user to choose the number of downloads (default is
1,000).</li>
     *     <li>Downloads the specified object from the directory bucket and measures
the total time.</li>
     *     <li>Downloads the same object from the regular bucket and measures the
total time.</li>
     *     <li>Compares the time differences and prints the results.</li>
     * </ul>
     *
     * <p>Note: The performance difference will be more pronounced if this example
is run on an EC2 instance
     * in the same Availability Zone as the buckets.
     *
     * @param bucketObject the name of the object to download
     */
    private static void demonstratePerformance(String bucketObject) {
        logger.info(DASHES);
        logger.info("6. Demonstrate the performance difference.");
        logger.info("""
            Now, let's do a performance test. We'll download the same object from
each
            bucket repeatedly and compare the total time needed.
```

```
            Note: the performance difference will be much more pronounced if this
            example is run in an EC2 instance in the same Availability Zone as
            the bucket.
            """);
        waitForInputToContinue(scanner);

        int downloads = 1000; // Default value.
        logger.info("The default number of downloads of the same object for this
example is set at " + downloads + ".");

        // Ask if the user wants to download a different number.
        logger.info("Would you like to download the file a different number of
times? (y/n): ");
        String response = scanner.next().trim().toLowerCase();
        if (response.equals("y")) {
            int maxDownloads = 1_000_000;

            // Ask for a valid number of downloads.
            while (true) {
                logger.info("Enter a number between 1 and " + maxDownloads + " for
the number of downloads: ");
                if (scanner.hasNextInt()) {
                    downloads = scanner.nextInt();
                    if (downloads >= 1 && downloads <= maxDownloads) {
                        break;
                    } else {
                        logger.info("Please enter a number between 1 and " +
maxDownloads + ".");
                    }
                } else {
                    logger.info("Invalid input. Please enter a valid integer.");
                    scanner.next();
                }
            }

            logger.info("You have chosen to download {}  items.", downloads);
        } else {
            logger.info("No changes made. Using default downloads: {}", downloads);
        }
        // Simulating the download process for the directory bucket.
        logger.info("Downloading from the directory bucket.");
        long directoryTimeStart = System.nanoTime();
        for (int index = 0; index < downloads; index++) {
            if (index % 50 == 0) {
```

```
                logger.info("Download " + index + " of " + downloads);
            }

            try {
                // Get the object from the directory bucket.
                s3DirectoriesActions.getObjectAsync(mS3ExpressClient,
mdirectoryBucketName, bucketObject).join();
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof NoSuchKeyException) {
                    logger.error("S3Exception occurred: {}", cause.getMessage(),
ce);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                }
                return;
            }
        }
        long directoryTimeDifference = System.nanoTime() - directoryTimeStart;

        // Download from the regular bucket.
        logger.info("Downloading from the regular bucket.");
        long normalTimeStart = System.nanoTime();
        for (int index = 0; index < downloads; index++) {
            if (index % 50 == 0) {
                logger.info("Download " + index + " of " + downloads);
            }

            try {
                s3DirectoriesActions.getObjectAsync(mS3RegularClient,
mregularBucketName, bucketObject).join();
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof NoSuchKeyException) {
                    logger.error("S3Exception occurred: {}", cause.getMessage(),
ce);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                }
                return;
            }
        }
```

```
        long normalTimeDifference = System.nanoTime() - normalTimeStart;
        logger.info("The directory bucket took " + directoryTimeDifference
+ " nanoseconds, while the regular bucket took " + normalTimeDifference + "
nanoseconds.");
        long difference = normalTimeDifference - directoryTimeDifference;
        logger.info("That's a difference of " + difference + " nanoseconds, or");
        logger.info(difference / 1_000_000_000.0 + " seconds.");

        if (difference < 0) {
            logger.info("The directory buckets were slower. This can happen if you
are not running on the cloud within a VPC.");
        }
        waitForInputToContinue(scanner);
    }

    private static String createSessionAddObjects() {
        logger.info(DASHES);
        logger.info("""
            5. Create an object and copy it.
            We'll create an object consisting of some text and upload it to the
            regular bucket.
            """);
        waitForInputToContinue(scanner);

        String bucketObject = "basic-text-object.txt";
        try {
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, bucketObject, "Look Ma, I'm a bucket!").join();
            s3DirectoriesActions.createSessionAsync(mS3ExpressClient,
mdirectoryBucketName).join();

            // Copy the object to the destination S3 bucket.
            s3DirectoriesActions.copyObjectAsync(mS3ExpressClient,
mregularBucketName, bucketObject, mdirectoryBucketName, bucketObject).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof S3Exception) {
                logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
        }
```

```
        logger.info("""
            It worked! This is because the S3Client that performed the copy
operation
            is the expressClient using the credentials for the user with permission
to
            work with directory buckets.

            It's important to remember the user permissions when interacting with
            directory buckets. Instead of validating permissions on every call as
            regular buckets do, directory buckets utilize the user credentials and
session
            token to validate. This allows for much faster connection speeds on
every call.
            For single calls, this is low, but for many concurrent calls
            this adds up to a lot of time saved.
            """);
        waitForInputToContinue(scanner);
        return bucketObject;
    }

    /**
     * Creates VPC users for the S3 Express One Zone scenario.
     * <p>
     * This method performs the following steps:
     * <ol>
     *     <li>Optionally creates a new VPC and VPC Endpoint if the application
is running in an EC2 instance in the same Availability Zone as the directory
buckets.</li>
     *     <li>Creates two IAM users: one with S3 Express One Zone permissions and
one without.</li>
     * </ol>
     *
     * @return a {@link UserNames} object containing the names of the created IAM
users
     */
    public static UserNames createVpcUsers() {
        /*
        Optionally create a VPC.
        Create two IAM users, one with S3 Express One Zone permissions and one
without.
        */
        logger.info(DASHES);
        logger.info("""
```

```
            1. First, we'll set up a new VPC and VPC Endpoint if this program is
running in an EC2 instance in the same AZ as your\s
            directory buckets will be. Are you running this in an EC2 instance
located in the same AZ as your intended directory buckets?
            """);

        logger.info("Do you want to setup a VPC Endpoint? (y/n)");
        String endpointAns = scanner.nextLine().trim();
        if (endpointAns.equalsIgnoreCase("y")) {
            logger.info("""
                Great! Let's set up a VPC, retrieve the Route Table from it, and
create a VPC Endpoint to connect the S3 Client to.
                """);
            try {
                s3DirectoriesActions.setupVPCAsync().join();
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof Ec2Exception) {
                    logger.error("IamException occurred: {}", cause.getMessage(),
ce);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                }
            }
            waitForInputToContinue(scanner);
        } else {
            logger.info("Skipping the VPC setup. Don't forget to use this in
production!");
        }
        logger.info(DASHES);
        logger.info("""
            2. Create a RegularUser and ExpressUser by using the AWS CDK.
            One IAM User, named RegularUser, will have permissions to work only
            with regular buckets and one IAM user, named ExpressUser, will have
            permissions to work only with directory buckets.
            """);
        waitForInputToContinue(scanner);

        // Create two users required for this scenario.
        Map<String, String> stackOutputs = createUsersUsingCDK(stackName);
        regularUser = stackOutputs.get("RegularUser");
        expressUser = stackOutputs.get("ExpressUser");
```

```
        UserNames names = new UserNames();
        names.setRegularUserName(regularUser);
        names.setExpressUserName(expressUser);
        return names;
    }

    /**
     * Creates users using AWS CloudFormation.
     *
     * @return a {@link Map} of String keys and String values representing the stack
outputs,
     * which may include user-related information such as user names and IDs.
     */
    public static Map<String, String> createUsersUsingCDK(String stackName) {
        logger.info("We'll use an AWS CloudFormation template to create the IAM
users and policies.");
        CloudFormationHelper.deployCloudFormationStack(stackName);
        return CloudFormationHelper.getStackOutputsAsync(stackName).join();
    }

    /**
     * Sets up the necessary clients and buckets for the S3 Express service.
     *
     * @param expressUserName the username for the user with S3 Express permissions
     * @param regularUserName the username for the user with regular S3 permissions
     */
    public static void setupClientsAndBuckets(String expressUserName, String
regularUserName) {
        Scanner locscanner = new Scanner(System.in);
        String accessKeyIdforRegUser;
        String secretAccessforRegUser;
        try {
            CreateAccessKeyResponse keyResponse =
s3DirectoriesActions.createAccessKeyAsync(regularUserName).join();
            accessKeyIdforRegUser = keyResponse.accessKey().accessKeyId();
            secretAccessforRegUser = keyResponse.accessKey().secretAccessKey();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IamException) {
                logger.error("IamException occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
```

```
            return;
        }

        String accessKeyIdforExpressUser;
        String secretAccessforExpressUser;
        try {
            CreateAccessKeyResponse keyResponseExpress =
s3DirectoriesActions.createAccessKeyAsync(expressUserName).join();
            accessKeyIdforExpressUser =
keyResponseExpress.accessKey().accessKeyId();
            secretAccessforExpressUser =
keyResponseExpress.accessKey().secretAccessKey();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IamException) {
                logger.error("IamException occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
            return;
        }

        logger.info(DASHES);
        logger.info("""
            3. Create two S3Clients; one uses the ExpressUser's credentials and one
uses the RegularUser's credentials.
            The 2 S3Clients will use different credentials.
            """);
        waitForInputToContinue(locscanner);
        try {
            mS3RegularClient =
createS3ClientWithAccessKeyAsync(accessKeyIdforRegUser,
secretAccessforRegUser).join();
            mS3ExpressClient =
createS3ClientWithAccessKeyAsync(accessKeyIdforExpressUser,
secretAccessforExpressUser).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IllegalArgumentException) {
                logger.error("An invalid argument exception occurred: {}",
cause.getMessage(), ce);
            } else {
```

```
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
            return;
        }

        logger.info("""
            We can now use the ExpressUser client to make calls to S3 Express
operations.
            """);
        waitForInputToContinue(locscanner);
        logger.info(DASHES);
        logger.info("""
            4. Create two buckets.
            Now we will create a directory bucket which is the linchpin of the S3
Express One Zone service. Directory buckets
            behave differently from regular S3 buckets which we will explore here.
We'll also create a regular bucket, put
            an object into the regular bucket, and copy it to the directory bucket.
            """);

        logger.info("""
            Now, let's choose an availability zone (AZ) for the directory bucket.
            We'll choose one that is supported.
            """);
        String zoneId;
        String regularBucketName;
        try {
            zoneId = s3DirectoriesActions.selectAvailabilityZoneIdAsync().join();
            regularBucketName = "reg-bucket-" + System.currentTimeMillis();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof Ec2Exception) {
                logger.error("EC2Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
            return;
        }
        logger.info("""
            Now, let's create the actual directory bucket, as well as a regular
bucket."
             """);
```

```
        String directoryBucketName = "test-bucket-" + System.currentTimeMillis() +
"--" + zoneId + "--x-s3";
        try {
            s3DirectoriesActions.createDirectoryBucketAsync(mS3ExpressClient,
directoryBucketName, zoneId).join();
            logger.info("Created directory bucket {}", directoryBucketName);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof BucketAlreadyExistsException) {
                logger.error("The bucket already exists. Moving on: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
                return;
            }
        }

        // Assign to the data member.
        mdirectoryBucketName = directoryBucketName;
        try {
            s3DirectoriesActions.createBucketAsync(mS3RegularClient,
regularBucketName).join();
            logger.info("Created regular bucket {} ", regularBucketName);
            mregularBucketName = regularBucketName;
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof BucketAlreadyExistsException) {
                logger.error("The bucket already exists. Moving on: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
                return;
            }
        }
        logger.info("Great! Both buckets were created.");
        waitForInputToContinue(locscanner);
    }

    /**
     * Creates an asynchronous S3 client with the specified access key and secret
access key.
```

```
     *
     * @param accessKeyId     the AWS access key ID
     * @param secretAccessKey the AWS secret access key
     * @return a {@link CompletableFuture} that asynchronously creates the S3 client
     * @throws IllegalArgumentException if the access key ID or secret access key is
 null
     */
    public static CompletableFuture<S3AsyncClient>
 createS3ClientWithAccessKeyAsync(String accessKeyId, String secretAccessKey) {
        return CompletableFuture.supplyAsync(() -> {
            // Validate input parameters
            if (accessKeyId == null || accessKeyId.isBlank() || secretAccessKey ==
 null || secretAccessKey.isBlank()) {
                throw new IllegalArgumentException("Access Key ID and Secret Access
 Key must not be null or empty");
            }

            AwsBasicCredentials awsCredentials =
 AwsBasicCredentials.create(accessKeyId, secretAccessKey);
            return S3AsyncClient.builder()

 .credentialsProvider(StaticCredentialsProvider.create(awsCredentials))
                .region(Region.US_WEST_2)
                .build();
        });
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Amazon S3 SDK 메서드의 래퍼 클래스입니다.

```
public class S3DirectoriesActions {

    private static IamAsyncClient iamAsyncClient;

    private static Ec2AsyncClient ec2AsyncClient;
    private static final Logger logger =
 LoggerFactory.getLogger(S3DirectoriesActions.class);

    private static IamAsyncClient getIAMAsyncClient() {
        if (iamAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            iamAsyncClient = IamAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return iamAsyncClient;
    }

    private static Ec2AsyncClient getEc2AsyncClient() {
        if (ec2AsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
```

```
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            ec2AsyncClient = Ec2AsyncClient.builder()
                .httpClient(httpClient)
                .region(Region.US_WEST_2)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return ec2AsyncClient;
    }

    /**
     * Deletes the specified S3 bucket and all the objects within it asynchronously.
     *
     * @param s3AsyncClient the S3 asynchronous client to use for the operations
     * @param bucketName the name of the S3 bucket to be deleted
     * @return a {@link CompletableFuture} that completes with a {@link
WaiterResponse} containing the
     *         {@link HeadBucketResponse} when the bucket has been successfully
deleted
     * @throws CompletionException if there was an error deleting the bucket or its
objects
     */
    public CompletableFuture<WaiterResponse<HeadBucketResponse>>
deleteBucketAndObjectsAsync(S3AsyncClient s3AsyncClient, String bucketName) {
        ListObjectsV2Request listRequest = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .build();

        return s3AsyncClient.listObjectsV2(listRequest)
            .thenCompose(listResponse -> {
                if (!listResponse.contents().isEmpty()) {
                    List<ObjectIdentifier> objectIdentifiers =
listResponse.contents().stream()
                        .map(s3Object ->
ObjectIdentifier.builder().key(s3Object.key()).build())
                        .collect(Collectors.toList());
```

```
                    DeleteObjectsRequest deleteRequest =
DeleteObjectsRequest.builder()
                        .bucket(bucketName)
                        .delete(Delete.builder().objects(objectIdentifiers).build())
                        .build();

                    return s3AsyncClient.deleteObjects(deleteRequest)
                        .thenAccept(deleteResponse -> {
                            if (!deleteResponse.errors().isEmpty()) {
                                deleteResponse.errors().forEach(error ->
                                    logger.error("Couldn't delete object " +
error.key() + ". Reason: " + error.message()));
                            }
                        });
                }
                return CompletableFuture.completedFuture(null);
            })
            .thenCompose(ignored -> {
                DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder()
                    .bucket(bucketName)
                    .build();
                return s3AsyncClient.deleteBucket(deleteBucketRequest);
            })
            .thenCompose(ignored -> {
                S3AsyncWaiter waiter = s3AsyncClient.waiter();
                HeadBucketRequest headBucketRequest =
HeadBucketRequest.builder().bucket(bucketName).build();
                return waiter.waitUntilBucketNotExists(headBucketRequest);
            })
            .whenComplete((ignored, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof S3Exception) {
                        throw new CompletionException("Error deleting bucket: " +
bucketName, cause);
                    }
                    throw new CompletionException("Failed to delete bucket and
objects: " + bucketName, exception);
                }
                logger.info("Bucket deleted successfully: " + bucketName);
            });
    }
```

```
   /**
    *  Lists the objects in an S3 bucket asynchronously.
    *
    * @param s3Client the S3 async client to use for the operation
    * @param bucketName the name of the S3 bucket containing the objects to list
    * @return a {@link CompletableFuture} that contains the list of object keys in
the specified bucket
    */
   public CompletableFuture<List<String>> listObjectsAsync(S3AsyncClient s3Client,
String bucketName) {
       ListObjectsV2Request request = ListObjectsV2Request.builder()
           .bucket(bucketName)
           .build();

       return s3Client.listObjectsV2(request)
           .thenApply(response -> response.contents().stream()
               .map(S3Object::key)
               .toList())
           .whenComplete((result, exception) -> {
               if (exception != null) {
                   throw new CompletionException("Couldn't list objects in bucket:
" + bucketName, exception);
               }
           });
   }

   /**
    * Retrieves an object from an Amazon S3 bucket asynchronously.
    *
    * @param s3Client   the S3 async client to use for the operation
    * @param bucketName the name of the S3 bucket containing the object
    * @param keyName    the unique identifier (key) of the object to retrieve
    * @return a {@link CompletableFuture} that, when completed, contains the
object's content as a {@link ResponseBytes} of {@link GetObjectResponse}
    */
   public CompletableFuture<ResponseBytes<GetObjectResponse>>
getObjectAsync(S3AsyncClient s3Client, String bucketName, String keyName) {
       GetObjectRequest objectRequest = GetObjectRequest.builder()
           .key(keyName)
           .bucket(bucketName)
           .build();

       // Get the object asynchronously and transform it into a byte array
```

```
        return s3Client.getObject(objectRequest, AsyncResponseTransformer.toBytes())
            .exceptionally(exception -> {
                Throwable cause = exception.getCause();
                if (cause instanceof NoSuchKeyException) {
                    throw new CompletionException("Failed to get the object. Reason:
" + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
                }
                throw new CompletionException("Failed to get the object",
exception);
            });
    }

    /**
     * Asynchronously copies an object from one S3 bucket to another.
     *
     * @param s3Client           the S3 async client to use for the copy operation
     * @param sourceBucket       the name of the source bucket
     * @param sourceKey          the key of the object to be copied in the source
bucket
     * @param destinationBucket  the name of the destination bucket
     * @param destinationKey     the key of the copied object in the destination
bucket
     * @return a {@link CompletableFuture} that completes when the copy operation is
finished
     */
    public CompletableFuture<Void> copyObjectAsync(S3AsyncClient s3Client, String
sourceBucket, String sourceKey, String destinationBucket, String destinationKey) {
        CopyObjectRequest copyRequest = CopyObjectRequest.builder()
            .sourceBucket(sourceBucket)
            .sourceKey(sourceKey)
            .destinationBucket(destinationBucket)
            .destinationKey(destinationKey)
            .build();

        return s3Client.copyObject(copyRequest)
            .thenRun(() -> logger.info("Copied object '" + sourceKey + "' from
bucket '" + sourceBucket + "' to bucket '" + destinationBucket + "'"))
            .whenComplete((ignored, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof S3Exception) {
                        throw new CompletionException("Couldn't copy object '" +
sourceKey + "' from bucket '" + sourceBucket + "' to bucket '" + destinationBucket
+ "'. Reason: " + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
```

```
                    }
                    throw new CompletionException("Failed to copy object",
exception);
                }
            });
    }

    /**
     * Asynchronously creates a session for the specified S3 bucket.
     *
     * @param s3Client   the S3 asynchronous client to use for creating the session
     * @param bucketName the name of the S3 bucket for which to create the session
     * @return a {@link CompletableFuture} that completes when the session is
created, or throws a {@link CompletionException} if an error occurs
     */
    public CompletableFuture<CreateSessionResponse> createSessionAsync(S3AsyncClient
s3Client, String bucketName) {
        CreateSessionRequest request = CreateSessionRequest.builder()
            .bucket(bucketName)
            .build();

        return s3Client.createSession(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof S3Exception) {
                        throw new CompletionException("Couldn't create the session.
Reason: " + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
                    }
                    throw new CompletionException("Unexpected error occurred while
creating session", exception);
                }
                logger.info("Created session for bucket: " + bucketName);
            });

    }

    /**
     * Creates a new S3 directory bucket in a specified Zone (For example, a
     * specified Availability Zone in this code example).
     *
     * @param s3Client   The asynchronous S3 client used to create the bucket
     * @param bucketName The name of the bucket to be created
     * @param zone       The Availability Zone where the bucket will be created
```

```
     * @throws CompletionException if there's an error creating the bucket
     */
    public CompletableFuture<CreateBucketResponse>
createDirectoryBucketAsync(S3AsyncClient s3Client, String bucketName, String zone)
{
        logger.info("Creating bucket: " + bucketName);

        CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
            .location(LocationInfo.builder()
                .type(LocationType.AVAILABILITY_ZONE)
                .name(zone)
                .build())
            .bucket(BucketInfo.builder()
                .type(BucketType.DIRECTORY)
                .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                .build())
            .build();

        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .createBucketConfiguration(bucketConfiguration)
            .build();

        return s3Client.createBucket(bucketRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof BucketAlreadyExistsException) {
                        throw new CompletionException("The bucket already exists: "
+ ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
                    }
                    throw new CompletionException("Unexpected error occurred while
creating bucket", exception);
                }
                logger.info("Bucket created successfully with location: " +
response.location());
            });
    }

    /**
     * Creates an S3 bucket asynchronously.
     *
     * @param s3Client    the S3 async client to use for the bucket creation
```

```
    * @param bucketName  the name of the S3 bucket to create
    * @return a {@link CompletableFuture} that completes with the {@link
WaiterResponse} containing the {@link HeadBucketResponse}
    *         when the bucket is successfully created
    * @throws CompletionException if there's an error creating the bucket
    */
   public CompletableFuture<WaiterResponse<HeadBucketResponse>>
createBucketAsync(S3AsyncClient s3Client, String bucketName) {
       CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
           .bucket(bucketName)
           .build();

       return s3Client.createBucket(bucketRequest)
           .thenCompose(response -> {
               S3AsyncWaiter s3Waiter = s3Client.waiter();
               HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                   .bucket(bucketName)
                   .build();
               return s3Waiter.waitUntilBucketExists(bucketRequestWait);
           })
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof BucketAlreadyExistsException) {
                       throw new CompletionException("The S3 bucket exists: " +
cause.getMessage(), cause);
                   } else {
                       throw new CompletionException("Failed to create access key:
" + exception.getMessage(), exception);
                   }
               }
               logger.info(bucketName + " is ready");
           });
   }

   /**
    * Uploads an object to an Amazon S3 bucket asynchronously.
    *
    * @param s3Client     the S3 async client to use for the upload
    * @param bucketName   the destination S3 bucket name
    * @param bucketObject the name of the object to be uploaded
    * @param text         the content to be uploaded as the object
    */
```

```
   public CompletableFuture<PutObjectResponse> putObjectAsync(S3AsyncClient
s3Client, String bucketName, String bucketObject, String text) {
       PutObjectRequest objectRequest = PutObjectRequest.builder()
           .bucket(bucketName)
           .key(bucketObject)
           .build();

       return s3Client.putObject(objectRequest, AsyncRequestBody.fromString(text))
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof NoSuchBucketException) {
                       throw new CompletionException("The S3 bucket does not exist:
" + cause.getMessage(), cause);
                   } else {
                       throw new CompletionException("Failed to create access key:
" + exception.getMessage(), exception);
                   }
               }
           });
   }

   /**
    * Creates an AWS IAM access key asynchronously for the specified user name.
    *
    * @param userName the name of the IAM user for whom to create the access key
    * @return a {@link CompletableFuture} that completes with the {@link
CreateAccessKeyResponse} containing the created access key
    */
   public CompletableFuture<CreateAccessKeyResponse> createAccessKeyAsync(String
userName) {
       CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
           .userName(userName)
           .build();

       return getIAMAsyncClient().createAccessKey(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   logger.info("Access Key Created.");
               } else {
                   if (exception == null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof IamException) {
```

```
                            throw new CompletionException("IAM error while creating
 access key: " + cause.getMessage(), cause);
                        } else {
                            throw new CompletionException("Failed to create access
 key: " + exception.getMessage(), exception);
                        }
                    }
                }
            });
    }

    /**
     * Asynchronously selects an Availability Zone ID from the available EC2 zones.
     *
     * @return A {@link CompletableFuture} that resolves to the selected
 Availability Zone ID.
     * @throws CompletionException if an error occurs during the request or
 processing.
     */
    public CompletableFuture<String> selectAvailabilityZoneIdAsync() {
        DescribeAvailabilityZonesRequest zonesRequest =
 DescribeAvailabilityZonesRequest.builder()
            .build();

        return getEc2AsyncClient().describeAvailabilityZones(zonesRequest)
            .thenCompose(response -> {
                List<AvailabilityZone> zonesList = response.availabilityZones();
                if (zonesList.isEmpty()) {
                    logger.info("No availability zones found.");
                    return CompletableFuture.completedFuture(null); // Return null
 if no zones are found
                }

                List<String> zoneIds = zonesList.stream()
                    .map(AvailabilityZone::zoneId) // Get the zoneId (e.g., "usw2-
az1")
                    .toList();

                return CompletableFuture.supplyAsync(() ->
 promptUserForZoneSelection(zonesList, zoneIds))
                    .thenApply(selectedZone -> {
                        // Return only the selected Zone ID (e.g., "usw2-az1").
                        return selectedZone.zoneId();
                    });
```

```
            })
            .whenComplete((result, exception) -> {
                if (exception == null) {
                    if (result != null) {
                        logger.info("Selected Availability Zone ID: " + result);
                    } else {
                        logger.info("No availability zone selected.");
                    }
                } else {
                    Throwable cause = exception.getCause();
                    if (cause instanceof Ec2Exception) {
                        throw new CompletionException("EC2 error while selecting
availability zone: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Failed to select availability
zone: " + exception.getMessage(), exception);
                }
            });
    }

    /**
     * Prompts the user to select an Availability Zone from the given list.
     *
     * @param zonesList the list of Availability Zones
     * @param zoneIds the list of zone IDs
     * @return the selected Availability Zone
     */
    private static AvailabilityZone
promptUserForZoneSelection(List<AvailabilityZone> zonesList, List<String> zoneIds)
{
        Scanner scanner = new Scanner(System.in);
        int index = -1;

        while (index < 0 || index >= zoneIds.size()) {
            logger.info("Select an availability zone:");
            IntStream.range(0, zoneIds.size()).forEach(i ->
                logger.info(i + ": " + zoneIds.get(i))
            );

            logger.info("Enter the number corresponding to your choice: ");
            if (scanner.hasNextInt()) {
                index = scanner.nextInt();
            } else {
                scanner.next();
```

```
            }
        }

        AvailabilityZone selectedZone = zonesList.get(index);
        logger.info("You selected: " + selectedZone.zoneId());
        return selectedZone;
    }

    /**
     * Asynchronously sets up a new VPC, including creating the VPC, finding the
associated route table, and
     * creating a VPC endpoint for the S3 service.
     *
     * @return a {@link CompletableFuture} that, when completed, contains a
AbstractMap with the
     *         VPC ID and VPC endpoint ID.
     */
    public CompletableFuture<AbstractMap.SimpleEntry<String, String>>
setupVPCAsync() {
        String cidr = "10.0.0.0/16";
        CreateVpcRequest vpcRequest = CreateVpcRequest.builder()
            .cidrBlock(cidr)
            .build();

        return getEc2AsyncClient().createVpc(vpcRequest)
            .thenCompose(vpcResponse -> {
                String vpcId = vpcResponse.vpc().vpcId();
                logger.info("VPC Created: {}", vpcId);

                Ec2AsyncWaiter waiter = getEc2AsyncClient().waiter();
                DescribeVpcsRequest request = DescribeVpcsRequest.builder()
                    .vpcIds(vpcId)
                    .build();

                return waiter.waitUntilVpcAvailable(request)
                    .thenApply(waiterResponse -> vpcId);
            })
            .thenCompose(vpcId -> {
                Filter filter = Filter.builder()
                    .name("vpc-id")
                    .values(vpcId)
                    .build();
```

```
                DescribeRouteTablesRequest describeRouteTablesRequest =
DescribeRouteTablesRequest.builder()
                    .filters(filter)
                    .build();

                return
getEc2AsyncClient().describeRouteTables(describeRouteTablesRequest)
                    .thenApply(routeTablesResponse -> {
                        if (routeTablesResponse.routeTables().isEmpty()) {
                            throw new CompletionException("No route tables found for
VPC: " + vpcId, null);
                        }
                        String routeTableId =
routeTablesResponse.routeTables().get(0).routeTableId();
                        logger.info("Route table found: {}", routeTableId);
                        return new AbstractMap.SimpleEntry<>(vpcId, routeTableId);
                    });
            })
            .thenCompose(vpcAndRouteTable -> {
                String vpcId = vpcAndRouteTable.getKey();
                String routeTableId = vpcAndRouteTable.getValue();
                Region region =
getEc2AsyncClient().serviceClientConfiguration().region();
                String serviceName = String.format("com.amazonaws.%s.s3express",
region.id());

                CreateVpcEndpointRequest endpointRequest =
CreateVpcEndpointRequest.builder()
                    .vpcId(vpcId)
                    .routeTableIds(routeTableId)
                    .serviceName(serviceName)
                    .build();

                return getEc2AsyncClient().createVpcEndpoint(endpointRequest)
                    .thenApply(vpcEndpointResponse -> {
                        String vpcEndpointId =
vpcEndpointResponse.vpcEndpoint().vpcEndpointId();
                        logger.info("VPC Endpoint created: {}", vpcEndpointId);
                        return new AbstractMap.SimpleEntry<>(vpcId, vpcEndpointId);
                    });
            })
            .exceptionally(exception -> {
                Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
```

```
                if (cause instanceof Ec2Exception) {
                    logger.error("EC2 error during VPC setup: {}",
 cause.getMessage(), cause);
                    throw new CompletionException("EC2 error during VPC setup: " +
 cause.getMessage(), cause);
                }

                logger.error("VPC setup failed: {}", cause.getMessage(), cause);
                throw new CompletionException("VPC setup failed: " +
 cause.getMessage(), cause);
            });
    }

}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 항목을 참조하세요.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObject
  - GetObject
  - ListObjects
  - PutObject

작업

`AbortMultipartUpload`

다음 코드 예시는 `AbortMultipartUpload`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷에서 멀티파트 업로드를 중단합니다.

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

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;



    /**
     * Aborts a specific multipart upload for the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be uploaded
     * @param uploadId   The upload ID of the multipart upload to abort
     * @return True if the multipart upload is successfully aborted, false otherwise
     */
    public static boolean abortDirectoryBucketMultipartUpload(S3Client s3Client,
 String bucketName,
            String objectKey, String uploadId) {
        logger.info("Aborting multipart upload: {} for bucket: {}", uploadId,
 bucketName);
        try {
            // Abort the multipart upload
            AbortMultipartUploadRequest abortMultipartUploadRequest =
 AbortMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .uploadId(uploadId)
                    .build();

            s3Client.abortMultipartUpload(abortMultipartUploadRequest);
            logger.info("Aborted multipart upload: {} for object: {}", uploadId,
 objectKey);
            return true;
        } catch (S3Exception e) {
```

```
            logger.error("Failed to abort multipart upload: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 AbortMultipartUpload를 참조하세요.

`CompleteMultipartUpload`

다음 코드 예시는 `CompleteMultipartUpload`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷에서 멀티파트 업로드를 완료합니다.

```
import com.example.s3.util.S3DirectoryBucketUtils;
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.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
```

```
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;


    /**
     * This method completes the multipart upload request by collating all the
     * upload parts.
     *
     * @param s3Client    The S3 client used to interact with S3
     * @param bucketName  The name of the directory bucket
     * @param objectKey   The key (name) of the object to be uploaded
     * @param uploadId    The upload ID used to track the multipart upload
     * @param uploadParts The list of completed parts
     * @return True if the multipart upload is successfully completed, false
     *         otherwise
     */
    public static boolean completeDirectoryBucketMultipartUpload(S3Client s3Client,
 String bucketName, String objectKey,
            String uploadId, List<CompletedPart> uploadParts) {
        try {
            CompletedMultipartUpload completedMultipartUpload =
 CompletedMultipartUpload.builder()
                    .parts(uploadParts)
                    .build();
            CompleteMultipartUploadRequest completeMultipartUploadRequest =
 CompleteMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .uploadId(uploadId)
                    .multipartUpload(completedMultipartUpload)
                    .build();

            CompleteMultipartUploadResponse response =
 s3Client.completeMultipartUpload(completeMultipartUploadRequest);
            logger.info("Multipart upload completed. ETag: {}", response.eTag());
            return true;
        } catch (S3Exception e) {
```

```
            logger.error("Failed to complete multipart upload: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CompleteMultipartUpload를 참조하세요.

`CopyObject`

다음 코드 예시는 `CopyObject`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷의 객체를 다른 디렉터리 버킷으로 복사합니다.

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

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Copies an object from one S3 general purpose bucket to one S3 directory
     * bucket.
     *
     * @param s3Client     The S3 client used to interact with S3
     * @param sourceBucket The name of the source bucket
     * @param objectKey    The key (name) of the object to be copied
     * @param targetBucket The name of the target bucket
     */
    public static void copyDirectoryBucketObject(S3Client s3Client, String
 sourceBucket, String objectKey,
            String targetBucket) {
        logger.info("Copying object: {} from bucket: {} to bucket: {}", objectKey,
 sourceBucket, targetBucket);

        try {
            // Create a CopyObjectRequest
            CopyObjectRequest copyReq = CopyObjectRequest.builder()
                    .sourceBucket(sourceBucket)
                    .sourceKey(objectKey)
                    .destinationBucket(targetBucket)
                    .destinationKey(objectKey)
                    .build();

            // Copy the object
            CopyObjectResponse copyRes = s3Client.copyObject(copyReq);
            logger.info("Successfully copied {} from bucket {} into bucket {}.
 CopyObjectResponse: {}",
                    objectKey, sourceBucket, targetBucket,
 copyRes.copyObjectResult().toString());

        } catch (S3Exception e) {
            logger.error("Failed to copy object: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CopyObject를 참조하세요.

`CreateBucket`

다음 코드 예시는 `CreateBucket`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

S3 디렉터리 버킷을 만듭니다.

```
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.model.BucketInfo;
import software.amazon.awssdk.services.s3.model.BucketType;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateBucketResponse;
import software.amazon.awssdk.services.s3.model.DataRedundancy;
import software.amazon.awssdk.services.s3.model.LocationInfo;
import software.amazon.awssdk.services.s3.model.LocationType;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * Creates a new S3 directory bucket in a specified Zone (For example, a
     * specified Availability Zone in this code example).
     *
     * @param s3Client   The S3 client used to create the bucket
     * @param bucketName The name of the bucket to be created
     * @param zone       The region where the bucket will be created
     * @throws S3Exception if there's an error creating the bucket
     */
    public static void createDirectoryBucket(S3Client s3Client, String bucketName,
 String zone) throws S3Exception {
```

```
        logger.info("Creating bucket: {}", bucketName);

        CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
                .location(LocationInfo.builder()
                        .type(LocationType.AVAILABILITY_ZONE)
                        .name(zone).build())
                .bucket(BucketInfo.builder()
                        .type(BucketType.DIRECTORY)
                        .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                        .build())
                .build();
        try {
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .createBucketConfiguration(bucketConfiguration).build();
            CreateBucketResponse response = s3Client.createBucket(bucketRequest);
            logger.info("Bucket created successfully with location: {}",
response.location());
        } catch (S3Exception e) {
            logger.error("Error creating bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateBucket을 참조하세요.

`CreateMultipartUpload`

다음 코드 예시는 `CreateMultipartUpload`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷에 멀티파트 업로드를 생성합니다.

```
import com.example.s3.util.S3DirectoryBucketUtils;
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.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * This method creates a multipart upload request that generates a unique upload
     * ID used to track
     * all the upload parts.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be uploaded
     * @return The upload ID used to track the multipart upload
     */
    public static String createDirectoryBucketMultipartUpload(S3Client s3Client,
 String bucketName, String objectKey) {
        logger.info("Creating multipart upload for object: {} in bucket: {}",
 objectKey, bucketName);

        try {
            // Create a CreateMultipartUploadRequest
            CreateMultipartUploadRequest createMultipartUploadRequest =
 CreateMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Initiate the multipart upload
            CreateMultipartUploadResponse response =
 s3Client.createMultipartUpload(createMultipartUploadRequest);
            String uploadId = response.uploadId();
            logger.info("Multipart upload initiated. Upload ID: {}", uploadId);
            return uploadId;
```

```
        } catch (S3Exception e) {
            logger.error("Failed to create multipart upload: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateMultipartUpload를 참조하세요.

`DeleteBucket`

다음 코드 예시는 `DeleteBucket`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

S3 디렉터리 버킷을 삭제합니다.

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

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;

    /**
     * Deletes the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
```

```
     * @param bucketName The name of the directory bucket to delete
     */
    public static void deleteDirectoryBucket(S3Client s3Client, String bucketName) {
        logger.info("Deleting bucket: {}", bucketName);

        try {
            // Create a DeleteBucketRequest
            DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Delete the bucket
            s3Client.deleteBucket(deleteBucketRequest);
            logger.info("Successfully deleted bucket: {}", bucketName);

        } catch (S3Exception e) {
            logger.error("Failed to delete bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteBucket을 참조하세요.

`DeleteBucketEncryption`

다음 코드 예시는 `DeleteBucketEncryption`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷의 암호화 구성을 삭제합니다.

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * Deletes the encryption configuration from an S3 bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     */
    public static void deleteDirectoryBucketEncryption(S3Client s3Client, String
 bucketName) {
        DeleteBucketEncryptionRequest deleteRequest =
 DeleteBucketEncryptionRequest.builder()
                .bucket(bucketName)
                .build();

        try {
            s3Client.deleteBucketEncryption(deleteRequest);
            logger.info("Bucket encryption deleted for bucket: {}", bucketName);
        } catch (S3Exception e) {
            logger.error("Failed to delete bucket encryption: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteBucketEncryption을 참조하세요.

`DeleteBucketPolicy`

다음 코드 예시는 `DeleteBucketPolicy`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷에 대한 버킷 정책을 삭제합니다.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketPolicy;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;


    /**
     * Deletes the bucket policy for the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     */
    public static void deleteDirectoryBucketPolicy(S3Client s3Client, String
 bucketName) {
        logger.info("Deleting policy for bucket: {}", bucketName);

        try {
            // Create a DeleteBucketPolicyRequest
            DeleteBucketPolicyRequest deletePolicyReq =
 DeleteBucketPolicyRequest.builder()
                    .bucket(bucketName)
                    .build();
```

```
            // Delete the bucket policy
            s3Client.deleteBucketPolicy(deletePolicyReq);
            logger.info("Successfully deleted bucket policy");

        } catch (S3Exception e) {
            logger.error("Failed to delete bucket policy: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteBucketPolicy를 참조하세요.

`DeleteObject`

다음 코드 예시는 `DeleteObject`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷의 객체를 삭제합니다.

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

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;




    /**
     * Deletes an object from the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be deleted
     */
    public static void deleteDirectoryBucketObject(S3Client s3Client, String
 bucketName, String objectKey) {
        logger.info("Deleting object: {} from bucket: {}", objectKey, bucketName);

        try {
            // Create a DeleteObjectRequest
            DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Delete the object
            s3Client.deleteObject(deleteObjectRequest);
            logger.info("Object {} has been deleted", objectKey);

        } catch (S3Exception e) {
            logger.error("Failed to delete object: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteObject를 참조하세요.

`DeleteObjects`

다음 코드 예시는 `DeleteObjects`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷에서 여러 객체를 삭제합니다.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectsResponse;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.net.URISyntaxException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;


    /**
     * Deletes multiple objects from the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKeys The list of keys (names) of the objects to be deleted
     */
    public static void deleteDirectoryBucketObjects(S3Client s3Client, String
 bucketName, List<String> objectKeys) {
```

```
        logger.info("Deleting objects from bucket: {}", bucketName);

        try {
            // Create a list of ObjectIdentifier.
            List<ObjectIdentifier> identifiers = objectKeys.stream()
                    .map(key -> ObjectIdentifier.builder().key(key).build())
                    .toList();

            Delete delete = Delete.builder()
                    .objects(identifiers)
                    .build();

            DeleteObjectsRequest deleteObjectsRequest =
DeleteObjectsRequest.builder()
                    .bucket(bucketName)
                    .delete(delete)
                    .build();

            DeleteObjectsResponse deleteObjectsResponse =
s3Client.deleteObjects(deleteObjectsRequest);
            deleteObjectsResponse.deleted().forEach(deleted -> logger.info("Deleted
object: {}", deleted.key()));

        } catch (S3Exception e) {
            logger.error("Failed to delete objects: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteObjects를 참조하세요.

`GetBucketEncryption`

다음 코드 예시는 `GetBucketEncryption`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷의 암호화 구성을 가져옵니다.

```
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.model.GetBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.GetBucketEncryptionResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionRule;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * Retrieves the encryption configuration for an S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @return The type of server-side encryption applied to the bucket (e.g.,
     *         AES256, aws:kms)
     */
    public static String getDirectoryBucketEncryption(S3Client s3Client, String
 bucketName) {
        try {
            // Create a GetBucketEncryptionRequest
            GetBucketEncryptionRequest getRequest =
 GetBucketEncryptionRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Retrieve the bucket encryption configuration
```

```
            GetBucketEncryptionResponse response =
s3Client.getBucketEncryption(getRequest);
            ServerSideEncryptionRule rule =
response.serverSideEncryptionConfiguration().rules().get(0);

            String encryptionType =
rule.applyServerSideEncryptionByDefault().sseAlgorithmAsString();
            logger.info("Bucket encryption algorithm: {}", encryptionType);
            logger.info("KMS Customer Managed Key ID: {}",
rule.applyServerSideEncryptionByDefault().kmsMasterKeyID());
            logger.info("Bucket Key Enabled: {}", rule.bucketKeyEnabled());

            return encryptionType;
        } catch (S3Exception e) {
            logger.error("Failed to get bucket encryption: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetBucketEncryption을 참조하세요.

`GetBucketPolicy`

다음 코드 예시는 `GetBucketPolicy`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷의 정책을 가져옵니다.

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

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketPolicy;

    /**
     * Retrieves the bucket policy for the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @return The bucket policy text
     */
    public static String getDirectoryBucketPolicy(S3Client s3Client, String
 bucketName) {
        logger.info("Getting policy for bucket: {}", bucketName);

        try {
            // Create a GetBucketPolicyRequest
            GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Retrieve the bucket policy
            GetBucketPolicyResponse response = s3Client.getBucketPolicy(policyReq);

            // Print and return the policy text
            String policyText = response.policy();
            logger.info("Bucket policy: {}", policyText);
            return policyText;

        } catch (S3Exception e) {
            logger.error("Failed to get bucket policy: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetBucketPolicy를 참조하세요.

`GetObject`

다음 코드 예시는 `GetObject`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷에서 객체를 가져옵니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
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.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.charset.StandardCharsets;
import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Retrieves an object from the specified S3 directory bucket.
```

```
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be retrieved
     * @return The retrieved object as a ResponseInputStream
     */
    public static boolean getDirectoryBucketObject(S3Client s3Client, String
bucketName, String objectKey) {
        logger.info("Retrieving object: {} from bucket: {}", objectKey, bucketName);

        try {
            // Create a GetObjectRequest
            GetObjectRequest objectRequest = GetObjectRequest.builder()
                    .key(objectKey)
                    .bucket(bucketName)
                    .build();

            // Retrieve the object as bytes
            ResponseBytes<GetObjectResponse> objectBytes =
s3Client.getObjectAsBytes(objectRequest);
            byte[] data = objectBytes.asByteArray();

            // Print object contents to console
            String objectContent = new String(data, StandardCharsets.UTF_8);
            logger.info("Object contents: \n{}", objectContent);

            return true;

        } catch (S3Exception e) {
            logger.error("Failed to retrieve object: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetObject를 참조하세요.

GetObjectAttributes

다음 코드 예시는 GetObjectAttributes의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷에서 객체 속성을 가져옵니다.

```
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.model.GetObjectAttributesRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAttributesResponse;
import software.amazon.awssdk.services.s3.model.ObjectAttributes;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Retrieves attributes for an object in the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to retrieve attributes for
     * @return True if the object attributes are successfully retrieved, false
     *         otherwise
     */
    public static boolean getDirectoryBucketObjectAttributes(S3Client s3Client,
 String bucketName, String objectKey) {
        logger.info("Retrieving attributes for object: {} from bucket: {}",
 objectKey, bucketName);
```

```
        try {
            // Create a GetObjectAttributesRequest
            GetObjectAttributesRequest getObjectAttributesRequest =
GetObjectAttributesRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .objectAttributes(ObjectAttributes.E_TAG,
ObjectAttributes.STORAGE_CLASS,
                            ObjectAttributes.OBJECT_SIZE)
                    .build();

            // Retrieve the object attributes
            GetObjectAttributesResponse response =
s3Client.getObjectAttributes(getObjectAttributesRequest);
            logger.info("Attributes for object {}:", objectKey);
            logger.info("ETag: {}", response.eTag());
            logger.info("Storage Class: {}", response.storageClass());
            logger.info("Object Size: {}", response.objectSize());
            return true;

        } catch (S3Exception e) {
            logger.error("Failed to retrieve object attributes: {} - Error code:
{}",
                    e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetObjectAttributes를 참조하세요.

HeadBucket

다음 코드 예시는 HeadBucket의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

지정된 S3 디렉터리 버킷이 존재하고 액세스할 수 있는지 확인합니다.

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

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * Checks if the specified S3 directory bucket exists and is accessible.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket to check
     * @return True if the bucket exists and is accessible, false otherwise
     */
    public static boolean headDirectoryBucket(S3Client s3Client, String bucketName)
 {
        logger.info("Checking if bucket exists: {}", bucketName);

        try {
            // Create a HeadBucketRequest
            HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();
            // If the bucket doesn't exist, the following statement throws
 NoSuchBucketException,
            // which is a subclass of S3Exception.
            s3Client.headBucket(headBucketRequest);
            logger.info("Amazon S3 directory bucket: \"{}\" found.", bucketName);
```

```
            return true;

        } catch (S3Exception e) {
            logger.error("Failed to access bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 HeadBucket을 참조하세요.

`HeadObject`

다음 코드 예시는 `HeadObject`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷에 있는 객체의 메타데이터를 가져옵니다.

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

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Retrieves metadata for an object in the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to retrieve metadata for
     * @return True if the object exists, false otherwise
     */
    public static boolean headDirectoryBucketObject(S3Client s3Client, String
 bucketName, String objectKey) {
        logger.info("Retrieving metadata for object: {} from bucket: {}", objectKey,
 bucketName);

        try {
            // Create a HeadObjectRequest
            HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Retrieve the object metadata
            HeadObjectResponse response = s3Client.headObject(headObjectRequest);
            logger.info("Amazon S3 object: \"{}\" found in bucket: \"{}\" with ETag:
 \"{}\"", objectKey, bucketName,
                    response.eTag());
            logger.info("Content-Type: {}", response.contentType());
            logger.info("Content-Length: {}", response.contentLength());
            logger.info("Last Modified: {}", response.lastModified());
            return true;

        } catch (S3Exception e) {
            logger.error("Failed to retrieve object metadata: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 HeadObject를 참조하세요.

`ListDirectoryBuckets`

다음 코드 예시는 `ListDirectoryBuckets`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

모든 디렉터리 버킷을 나열합니다.

```
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.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.List;
import java.util.UUID;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

     /**
     * Lists all S3 directory buckets and no general purpose buckets.
     *
     * @param s3Client The S3 client used to interact with S3
     * @return A list of bucket names
     */
    public static List<String> listDirectoryBuckets(S3Client s3Client) {
        logger.info("Listing all directory buckets");

        try {
            // Create a ListBucketsRequest
```

```
            ListDirectoryBucketsRequest listDirectoryBucketsRequest =
ListDirectoryBucketsRequest.builder().build();

            // Retrieve the list of buckets
            ListDirectoryBucketsResponse response =
s3Client.listDirectoryBuckets(listDirectoryBucketsRequest);

            // Extract bucket names
            List<String> bucketNames = response.buckets().stream()
                    .map(Bucket::name)
                    .collect(Collectors.toList());

            return bucketNames;
        } catch (S3Exception e) {
            logger.error("Failed to list buckets: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListDirectoryBuckets을 참조하세요.

`ListMultipartUploads`

다음 코드 예시는 `ListMultipartUploads`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷에 있는 멀티파트 업로드를 나열합니다.

```
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.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.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static
 com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;


    /**
     * Lists multipart uploads for the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @return A list of MultipartUpload objects representing the multipart uploads
     */
    public static List<MultipartUpload> listDirectoryBucketMultipartUploads(S3Client
 s3Client, String bucketName) {
        logger.info("Listing in-progress multipart uploads for bucket: {}",
 bucketName);

        try {
            // Create a ListMultipartUploadsRequest
            ListMultipartUploadsRequest listMultipartUploadsRequest =
 ListMultipartUploadsRequest.builder()
                    .bucket(bucketName)
                    .build();

            // List the multipart uploads
            ListMultipartUploadsResponse response =
 s3Client.listMultipartUploads(listMultipartUploadsRequest);
```

```
            List<MultipartUpload> uploads = response.uploads();
            for (MultipartUpload upload : uploads) {
                logger.info("In-progress multipart upload: Upload ID: {}, Key: {},
 Initiated: {}", upload.uploadId(),
                            upload.key(), upload.initiated());
            }
            return uploads;

        } catch (S3Exception e) {
            logger.error("Failed to list multipart uploads: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
            return List.of(); // Return an empty list if an exception is thrown
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListMultipartUploads를 참조하세요.

`ListObjectsV2`

다음 코드 예시는 `ListObjectsV2`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷에 있는 객체를 나열합니다.

```
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.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;
```

```
import java.nio.file.Path;
import java.util.List;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Lists objects in the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @return A list of object keys in the bucket
     */
    public static List<String> listDirectoryBucketObjectsV2(S3Client s3Client,
 String bucketName) {
        logger.info("Listing objects in bucket: {}", bucketName);

        try {
            // Create a ListObjectsV2Request
            ListObjectsV2Request listObjectsV2Request =
 ListObjectsV2Request.builder()
                    .bucket(bucketName)
                    .build();

            // Retrieve the list of objects
            ListObjectsV2Response response =
 s3Client.listObjectsV2(listObjectsV2Request);

            // Extract and return the object keys
            return response.contents().stream()
                    .map(S3Object::key)
                    .collect(Collectors.toList());

        } catch (S3Exception e) {
            logger.error("Failed to list objects: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
```

```
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListObjectsV2를 참조하세요.

`ListParts`

다음 코드 예시는 `ListParts`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷에 있는 멀티파트 업로드의 일부를 나열합니다.

```
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.model.ListPartsRequest;
import software.amazon.awssdk.services.s3.model.ListPartsResponse;
import software.amazon.awssdk.services.s3.model.Part;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static
 com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

   /**
    * Lists the parts of a multipart upload for the specified S3 directory bucket.
    *
    * @param s3Client   The S3 client used to interact with S3
    * @param bucketName The name of the directory bucket
    * @param objectKey  The key (name) of the object being uploaded
    * @param uploadId   The upload ID used to track the multipart upload
    * @return A list of Part representing the parts of the multipart upload
    */
   public static List<Part> listDirectoryBucketMultipartUploadParts(S3Client
 s3Client, String bucketName,
           String objectKey, String uploadId) {
       logger.info("Listing parts for object: {} in bucket: {}", objectKey,
 bucketName);

       try {
           // Create a ListPartsRequest
           ListPartsRequest listPartsRequest = ListPartsRequest.builder()
                   .bucket(bucketName)
                   .uploadId(uploadId)
                   .key(objectKey)
                   .build();

           // List the parts of the multipart upload
           ListPartsResponse response = s3Client.listParts(listPartsRequest);
           List<Part> parts = response.parts();
           for (Part part : parts) {
               logger.info("Uploaded part: Part number = \"{}\", etag = {}",
 part.partNumber(), part.eTag());
           }
           return parts;

       } catch (S3Exception e) {
           logger.error("Failed to list parts: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                   e.awsErrorDetails().errorCode());
           return List.of(); // Return an empty list if an exception is thrown
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListParts를 참조하세요.

`PutBucketEncryption`

다음 코드 예시는 `PutBucketEncryption`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷에 버킷 암호화를 설정합니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.ServerSideEncryption;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionByDefault;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionConfiguration;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionRule;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createKmsClient;
import static com.example.s3.util.S3DirectoryBucketUtils.createKmsKey;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.scheduleKeyDeletion;

    /**
     * Sets the default encryption configuration for an S3 bucket as SSE-KMS.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
```

```
    * @param kmsKeyId   The ID of the customer-managed KMS key
    */
   public static void putDirectoryBucketEncryption(S3Client s3Client, String
bucketName, String kmsKeyId) {
       // Define the default encryption configuration to use SSE-KMS. For directory
       // buckets, AWS managed KMS keys aren't supported. Only customer-managed
keys
       // are supported.
       ServerSideEncryptionByDefault encryptionByDefault =
ServerSideEncryptionByDefault.builder()
               .sseAlgorithm(ServerSideEncryption.AWS_KMS)
               .kmsMasterKeyID(kmsKeyId)
               .build();

       // Create a server-side encryption rule to apply the default encryption
       // configuration. For directory buckets, the bucketKeyEnabled field is
enforced
       // to be true.
       ServerSideEncryptionRule rule = ServerSideEncryptionRule.builder()
               .bucketKeyEnabled(true)
               .applyServerSideEncryptionByDefault(encryptionByDefault)
               .build();

       // Create the server-side encryption configuration for the bucket
       ServerSideEncryptionConfiguration encryptionConfiguration =
ServerSideEncryptionConfiguration.builder()
               .rules(rule)
               .build();

       // Create the PutBucketEncryption request
       PutBucketEncryptionRequest putRequest = PutBucketEncryptionRequest.builder()
               .bucket(bucketName)
               .serverSideEncryptionConfiguration(encryptionConfiguration)
               .build();

       // Set the bucket encryption
       try {
           s3Client.putBucketEncryption(putRequest);
           logger.info("SSE-KMS Bucket encryption configuration set for the
directory bucket: {}", bucketName);
       } catch (S3Exception e) {
           logger.error("Failed to set bucket encryption: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                   e.awsErrorDetails().errorCode());
```

```
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutBucketEncryption을 참조하세요.

`PutBucketPolicy`

다음 코드 예시는 `PutBucketPolicy`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷에 버킷 정책을 적용합니다.

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

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;

    /**
     * Sets the following bucket policy for the specified S3 directory bucket.
     *<pre>
     * {
     *     "Version":"2012-10-17",
     *     "Statement": [
     *         {
```

```
     *             "Sid": "AdminPolicy",
     *             "Effect": "Allow",
     *             "Principal": {
     *                 "AWS": "arn:aws:iam::<ACCOUNT_ID>:root"
     *             },
     *             "Action": "s3express:*",
     *             "Resource": "arn:aws:s3express:us-west-2:<ACCOUNT_ID>:bucket/
<DIR_BUCKET_NAME>
     *         }
     *     ]
     * }
     * </pre>
     * This policy grants all S3 directory bucket actions to identities in the same
 account as the bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param policyText The policy text to be applied
     */
    public static void putDirectoryBucketPolicy(S3Client s3Client, String
 bucketName, String policyText) {
        logger.info("Setting policy on bucket: {}", bucketName);
        logger.info("Policy: {}", policyText);

        try {
            PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
                    .bucket(bucketName)
                    .policy(policyText)
                    .build();

            s3Client.putBucketPolicy(policyReq);
            logger.info("Bucket policy set successfully!");

        } catch (S3Exception e) {
            logger.error("Failed to set bucket policy: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutBucketPolicy를 참조하세요.

PutObject

다음 코드 예시는 PutObject의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷에 객체를 업로드합니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.awscore.exception.AwsErrorDetails;
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.UncheckedIOException;
import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;

    /**
     * Puts an object into the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be placed in the bucket
     * @param filePath   The path of the file to be uploaded
     */
    public static void putDirectoryBucketObject(S3Client s3Client, String
 bucketName, String objectKey, Path filePath) {
```

```
        logger.info("Putting object: {} into bucket: {}", objectKey, bucketName);

        try {
            // Create a PutObjectRequest
            PutObjectRequest putObj = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Upload the object
            s3Client.putObject(putObj, filePath);
            logger.info("Successfully placed {} into bucket {}", objectKey,
bucketName);

        } catch (UncheckedIOException e) {
            throw S3Exception.builder().message("Failed to read the file: " +
e.getMessage()).cause(e)
                    .awsErrorDetails(AwsErrorDetails.builder()
                            .errorCode("ClientSideException:FailedToReadFile")
                            .errorMessage(e.getMessage())
                            .build())
                    .build();
        } catch (S3Exception e) {
            logger.error("Failed to put object: {}", e.getMessage(), e);
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 PutObject를 참조하세요.

`UploadPart`

다음 코드 예시는 `UploadPart`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

디렉터리 버킷에 멀티파트 업로드의 일부를 업로드합니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
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.CompletedPart;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.nio.ByteBuffer;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.List;

import static
 com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;

    /**
     * This method creates part requests and uploads individual parts to S3.
     * While it uses the UploadPart API to upload a single part, it does so
     * sequentially to handle multiple parts of a file, returning all the completed
     * parts.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be uploaded
     * @param uploadId   The upload ID used to track the multipart upload
     * @param filePath   The path to the file to be uploaded
     * @return A list of uploaded parts
     * @throws IOException if an I/O error occurs
```

```
     */
    public static List<CompletedPart> multipartUploadForDirectoryBucket(S3Client
s3Client, String bucketName,
            String objectKey, String uploadId, Path filePath) throws IOException {
        logger.info("Uploading parts for object: {} in bucket: {}", objectKey,
bucketName);

        int partNumber = 1;
        List<CompletedPart> uploadedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

        // Read the local file, break down into chunks and process
        try (RandomAccessFile file = new RandomAccessFile(filePath.toFile(), "r")) {
            long fileSize = file.length();
            int position = 0;

            // Sequentially upload parts of the file
            while (position < fileSize) {
                file.seek(position);
                int read = file.getChannel().read(bb);

                bb.flip(); // Swap position and limit before reading from the buffer
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                        .bucket(bucketName)
                        .key(objectKey)
                        .uploadId(uploadId)
                        .partNumber(partNumber)
                        .build();

                UploadPartResponse partResponse = s3Client.uploadPart(
                        uploadPartRequest,
                        RequestBody.fromByteBuffer(bb));

                // Build the uploaded part
                CompletedPart uploadedPart = CompletedPart.builder()
                        .partNumber(partNumber)
                        .eTag(partResponse.eTag())
                        .build();

                // Add the uploaded part to the list
                uploadedParts.add(uploadedPart);

                // Log to indicate the part upload is done
```

```
                logger.info("Uploaded part number: {} with ETag: {}", partNumber,
partResponse.eTag());

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (S3Exception e) {
            logger.error("Failed to list parts: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
            throw e;
        }
        return uploadedParts;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 UploadPart를 참조하세요.

`UploadPartCopy`

다음 코드 예시는 `UploadPartCopy`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

소스 객체 크기를 기반으로 복사할 부분을 나눈 다음 각 부분을 디렉터리 버킷에 복사합니다.

```
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.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
```

```
import software.amazon.awssdk.services.s3.model.UploadPartCopyRequest;
import software.amazon.awssdk.services.s3.model.UploadPartCopyResponse;

import java.io.IOException;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.List;

import static
 com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static
 com.example.s3.util.S3DirectoryBucketUtils.completeDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

    /**
     * Creates copy parts based on source object size and copies over individual
     * parts.
     *
     * @param s3Client          The S3 client used to interact with S3
     * @param sourceBucket      The name of the source bucket
     * @param sourceKey         The key (name) of the source object
     * @param destinationBucket The name of the destination bucket
     * @param destinationKey    The key (name) of the destination object
     * @param uploadId          The upload ID used to track the multipart upload
     * @return A list of completed parts
     */
    public static List<CompletedPart> multipartUploadCopyForDirectoryBucket(S3Client
 s3Client, String sourceBucket,
            String sourceKey, String destinationBucket, String destinationKey,
 String uploadId) {
        // Get the object size to track the end of the copy operation
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(sourceBucket)
                .key(sourceKey)
                .build();
```

```
        HeadObjectResponse headObjectResponse =
s3Client.headObject(headObjectRequest);
        long objectSize = headObjectResponse.contentLength();

        logger.info("Source Object size: {}", objectSize);

        // Copy the object using 20 MB parts
        long partSize = 20 * 1024 * 1024; // 20 MB
        long bytePosition = 0;
        int partNum = 1;
        List<CompletedPart> uploadedParts = new ArrayList<>();

        while (bytePosition < objectSize) {
            long lastByte = Math.min(bytePosition + partSize - 1, objectSize - 1);
            logger.info("Part Number: {}, Byte Position: {}, Last Byte: {}",
partNum, bytePosition, lastByte);

            try {
                UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
                        .sourceBucket(sourceBucket)
                        .sourceKey(sourceKey)
                        .destinationBucket(destinationBucket)
                        .destinationKey(destinationKey)
                        .uploadId(uploadId)
                        .copySourceRange("bytes=" + bytePosition + "-" + lastByte)
                        .partNumber(partNum)
                        .build();
                UploadPartCopyResponse uploadPartCopyResponse =
s3Client.uploadPartCopy(uploadPartCopyRequest);

                CompletedPart part = CompletedPart.builder()
                        .partNumber(partNum)
                        .eTag(uploadPartCopyResponse.copyPartResult().eTag())
                        .build();
                uploadedParts.add(part);

                bytePosition += partSize;
                partNum++;
            } catch (S3Exception e) {
                logger.error("Failed to copy part number {}: {} - Error code: {}",
partNum,
                        e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode());
```

```
                throw e;
            }
        }

        return uploadedParts;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 UploadPartCopy를 참조하세요.

시나리오

미리 서명된 URL을 만들어 객체 가져오기

다음 코드 예제에서는 S3 디렉터리 버킷에 대해 미리 서명된 URL을 만들고 객체를 가져오는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

S3 디렉터리 버킷의 객체에 액세스하기 위해 미리 서명된 GET URL을 생성합니다.

```
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.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 java.nio.file.Path;
import java.time.Duration;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Presigner;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;


   /**
    * Generates a presigned URL for accessing an object in the specified S3
    * directory bucket.
    *
    * @param s3Presigner The S3 presigner client used to generate the presigned URL
    * @param bucketName  The name of the directory bucket
    * @param objectKey   The key (name) of the object to access
    * @return A presigned URL for accessing the specified object
    */
   public static String generatePresignedGetURLForDirectoryBucket(S3Presigner
 s3Presigner, String bucketName,
           String objectKey) {
       logger.info("Generating presigned URL for object: {} in bucket: {}",
 objectKey, bucketName);

       try {
           // Create a GetObjectRequest
           GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                   .bucket(bucketName)
                   .key(objectKey)
                   .build();

           // Create a GetObjectPresignRequest
           GetObjectPresignRequest getObjectPresignRequest =
 GetObjectPresignRequest.builder()
                   .signatureDuration(Duration.ofMinutes(10)) // Presigned URL
 valid for 10 minutes
                   .getObjectRequest(getObjectRequest)
                   .build();

           // Generate the presigned URL
           PresignedGetObjectRequest presignedGetObjectRequest =
 s3Presigner.presignGetObject(getObjectPresignRequest);

           // Get the presigned URL
```

```
            String presignedURL = presignedGetObjectRequest.url().toString();
            logger.info("Presigned URL: {}", presignedURL);
            return presignedURL;

        } catch (S3Exception e) {
            logger.error("Failed to generate presigned URL: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetObject를 참조하세요.

SDK for Java 2.x를 사용한 SageMaker AI 예제

다음 코드 예제에서는 SageMaker AI와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시작하기
- 작업
- 시나리오

시작하기

SageMaker AI 시작

다음 코드 예제에서는 SageMaker AI 사용을 시작하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * 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);
        }
```

```
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListNotebookInstances를 참조하세요.

작업

`CreatePipeline`

다음 코드 예시는 `CreatePipeline`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreatePipeline을 참조하세요.

`DeletePipeline`

다음 코드 예시는 `DeletePipeline`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    // 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);
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeletePipeline을 참조하세요.

`DescribePipelineExecution`

다음 코드 예시는 `DescribePipelineExecution`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   // 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);
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribePipelineExecution을 참조하세요.

`StartPipelineExecution`

다음 코드 예시는 `StartPipelineExecution`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    // 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();
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 StartPipelineExecution을 참조하세요.

시나리오

지리공간 작업 및 파이프라인으로 시작하기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 파이프라인의 리소스를 설정하세요.
- 지리 공간 작업을 실행하는 파이프라인을 설정합니다.
- 파이프라인 실행을 시작합니다.
- 실행 상태를 모니터링합니다.

- 파이프라인의 출력을 볼 수 있습니다.
- 리소스를 정리합니다.

자세한 내용은 Community. AWS SDKs를 사용하여 SageMaker 파이프라인 생성 및 실행을 참조하세요.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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;
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 항목을 참조하세요.
  - CreatePipeline
  - DeletePipeline
  - DescribePipelineExecution
  - StartPipelineExecution
  - UpdatePipeline

Java 2.x용 SDK를 사용하는 Secrets Manager 예제

다음 코드 예제에서는 Secrets Manager와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`GetSecretValue`

다음 코드 예시는 `GetSecretValue`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetSecretValue를 참조하세요.

Java 2.x용 SDK를 사용하는 SES 예제

다음 코드 예제에서는 AWS SDK for Java 2.x Amazon SES에서를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`ListIdentities`

다음 코드 예시는 `ListIdentities`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListIdentities를 참조하세요.

`ListTemplates`

다음 코드 예시는 `ListTemplates`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListTemplates를 참조하세요.

`SendEmail`

다음 코드 예시는 `SendEmail`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import software.amazon.awssdk.services.ses.model.Content;
import software.amazon.awssdk.services.ses.model.Destination;
import software.amazon.awssdk.services.ses.model.Message;
import software.amazon.awssdk.services.ses.model.Body;
import software.amazon.awssdk.services.ses.model.SendEmailRequest;
import software.amazon.awssdk.services.ses.model.SesException;

import javax.mail.MessagingException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessageEmailRequest {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <sender> <recipient> <subject>\s

                Where:
                    sender - An email address that represents the sender.\s
                    recipient -  An email address that represents the recipient.\s
                    subject - The  subject line.\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];

        Region region = Region.US_EAST_1;
        SesClient client = SesClient.builder()
```

```
                .region(region)
                .build();

        // The HTML body of the email.
        String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</h1>"
                + "<p> See the list of customers.</p>" + "</body>" + "</html>";

        try {
            send(client, sender, recipient, subject, bodyHTML);
            client.close();
            System.out.println("Done");

        } catch (MessagingException e) {
            e.printStackTrace();
        }
    }

    public static void send(SesClient client,
            String sender,
            String recipient,
            String subject,
            String bodyHTML) throws MessagingException {

        Destination destination = Destination.builder()
                .toAddresses(recipient)
                .build();

        Content content = Content.builder()
                .data(bodyHTML)
                .build();

        Content sub = Content.builder()
                .data(subject)
                .build();

        Body body = Body.builder()
                .html(content)
                .build();

        Message msg = Message.builder()
                .subject(sub)
                .body(body)
                .build();
```

```
        SendEmailRequest emailRequest = SendEmailRequest.builder()
                .destination(destination)
                .message(msg)
                .source(sender)
                .build();

        try {
            System.out.println("Attempting to send an email through Amazon SES " +
 "using the AWS SDK for Java...");
            client.sendEmail(emailRequest);

        } catch (SesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import javax.activation.DataHandler;
import javax.activation.DataSource;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Session;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;
import javax.mail.internet.MimeBodyPart;
import javax.mail.util.ByteArrayDataSource;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.file.Files;
import java.util.Properties;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.ses.model.SendRawEmailRequest;
import software.amazon.awssdk.services.ses.model.RawMessage;
import software.amazon.awssdk.services.ses.model.SesException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class SendMessageAttachment {
    public static void main(String[] args) throws IOException {
        final String usage = """

                Usage:
                    <sender> <recipient> <subject> <fileLocation>\s

                Where:
                    sender - An email address that represents the sender.\s
                    recipient -  An email address that represents the recipient.\s
                    subject - The  subject line.\s
                    fileLocation - The location of a Microsoft Excel file to use as
 an attachment (C:/AWS/customers.xls).\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];
        String fileLocation = args[3];

        // The email body for recipients with non-HTML email clients.
        String bodyText = "Hello,\r\n" + "Please see the attached file for a list "
                + "of customers to contact.";

        // The HTML body of the email.
        String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</h1>"
                + "<p>Please see the attached file for a " + "list of customers to
 contact.</p>" + "</body>"
                + "</html>";

        Region region = Region.US_WEST_2;
        SesClient client = SesClient.builder()
                .region(region)
```

```
                .build();

        try {
            sendemailAttachment(client, sender, recipient, subject, bodyText,
bodyHTML, fileLocation);
            client.close();
            System.out.println("Done");

        } catch (IOException | MessagingException e) {
            e.printStackTrace();
        }
    }

    public static void sendemailAttachment(SesClient client,
            String sender,
            String recipient,
            String subject,
            String bodyText,
            String bodyHTML,
            String fileLocation) throws AddressException, MessagingException,
IOException {

        java.io.File theFile = new java.io.File(fileLocation);
        byte[] fileContent = Files.readAllBytes(theFile.toPath());

        Session session = Session.getDefaultInstance(new Properties());

        // Create a new MimeMessage object.
        MimeMessage message = new MimeMessage(session);

        // Add subject, from and to lines.
        message.setSubject(subject, "UTF-8");
        message.setFrom(new InternetAddress(sender));
        message.setRecipients(Message.RecipientType.TO,
InternetAddress.parse(recipient));

        // Create a multipart/alternative child container.
        MimeMultipart msgBody = new MimeMultipart("alternative");

        // Create a wrapper for the HTML and text parts.
        MimeBodyPart wrap = new MimeBodyPart();

        // Define the text part.
        MimeBodyPart textPart = new MimeBodyPart();
```

```
        textPart.setContent(bodyText, "text/plain; charset=UTF-8");

        // Define the HTML part.
        MimeBodyPart htmlPart = new MimeBodyPart();
        htmlPart.setContent(bodyHTML, "text/html; charset=UTF-8");

        // Add the text and HTML parts to the child container.
        msgBody.addBodyPart(textPart);
        msgBody.addBodyPart(htmlPart);

        // Add the child container to the wrapper object.
        wrap.setContent(msgBody);

        // Create a multipart/mixed parent container.
        MimeMultipart msg = new MimeMultipart("mixed");

        // Add the parent container to the message.
        message.setContent(msg);
        msg.addBodyPart(wrap);

        // Define the attachment.
        MimeBodyPart att = new MimeBodyPart();
        DataSource fds = new ByteArrayDataSource(fileContent,
                "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet");
        att.setDataHandler(new DataHandler(fds));

        String reportName = "WorkReport.xls";
        att.setFileName(reportName);

        // Add the attachment to the message.
        msg.addBodyPart(att);

        try {
            System.out.println("Attempting to send an email through Amazon SES " +
 "using the AWS SDK for Java...");

            ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
            message.writeTo(outputStream);

            ByteBuffer buf = ByteBuffer.wrap(outputStream.toByteArray());

            byte[] arr = new byte[buf.remaining()];
            buf.get(arr);
```

```
            SdkBytes data = SdkBytes.fromByteArray(arr);
            RawMessage rawMessage = RawMessage.builder()
                    .data(data)
                    .build();

            SendRawEmailRequest rawEmailRequest = SendRawEmailRequest.builder()
                    .rawMessage(rawMessage)
                    .build();

            client.sendRawEmail(rawEmailRequest);

        } catch (SesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Email sent using SesClient with attachment");
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 SendEmail을 참조하세요.

`SendTemplatedEmail`

다음 코드 예시는 `SendTemplatedEmail`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 SendTemplatedEmail을 참조하세요.

시나리오

DynamoDB 데이터를 추적하는 웹 애플리케이션 만들기

다음 코드 예제에서는 Amazon DynamoDB 테이블에서 작업 항목을 추적하고 Amazon Simple Email Service(Amazon SES)를 사용하여 보고서를 보내는 웹 애플리케이션을 만드는 방법을 보여줍니다.

SDK for Java 2.x

Amazon DynamoDB API를 사용하여 DynamoDB 작업 데이터를 추적하는 동적 웹 애플리케이션을 만드는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- DynamoDB
- Amazon SES

Amazon Redshift 데이터를 추적하는 웹 애플리케이션 만들기

다음 코드 예제에서는 Amazon Redshift 데이터베이스를 사용하는 작업 항목을 추적하고 보고하는 웹 애플리케이션을 만드는 방법을 보여줍니다.

SDK for Java 2.x

Amazon Redshift 데이터베이스에 저장된 작업 항목을 추적하고 보고하는 웹 애플리케이션을 만드는 방법을 보여줍니다.

Amazon Redshift 데이터를 쿼리하고 React 애플리케이션에서 사용하도록 Spring REST API를 설정하는 방법에 대한 지침과 전체 소스 코드는 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Redshift
- Amazon SES

Aurora 서버리스 작업 항목 트래커 만들기

다음 코드 예제에서는 Amazon Aurora Serverless 데이터베이스에서 작업 항목을 추적하고 Amazon Simple Email Service(Amazon SES)를 사용하여 보고서를 보내는 웹 애플리케이션을 만드는 방법을 보여줍니다.

SDK for Java 2.x

Amazon RDS 데이터베이스에 저장된 작업 항목을 추적하고 보고하는 웹 애플리케이션을 만드는 방법을 보여줍니다.

Amazon Aurora Serverless 데이터를 쿼리하고 React 애플리케이션에서 사용하도록 Spring REST API를 설정하는 방법에 대한 지침과 전체 소스 코드는 GitHub에서 전체 예제를 참조하세요.

JDBC API를 사용한 예제를 설정하고 실행하는 방법에 대한 전체 소스 코드와 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Aurora
- Amazon RDS
- Amazon RDS 데이터 서비스
- Amazon SES

이미지에서 PPE 감지

다음 코드 예제에서는 Amazon Rekognition을 사용하여 이미지에서 개인 보호 장비(PPE)를 감지하는 앱을 구축하는 방법을 보여줍니다.

SDK for Java 2.x

개인 보호 장비로 이미지를 감지하는 AWS Lambda 함수를 생성하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- DynamoDB
- Amazon Rekognition
- Amazon S3

- Amazon SES

이미지에서 객체 감지

다음 코드 예제에서는 Amazon Rekognition을 사용하여 이미지에서 범주별 객체를 감지하는 앱을 구축하는 방법을 보여줍니다.

SDK for Java 2.x

Amazon Rekognition을 사용하여 Amazon Simple Storage Service(Amazon S3) 버킷에 있는 이미지에서 범주별로 객체를 식별하기 위해 Amazon Rekognition을 사용하여 앱을 만드는 방법을 보여줍니다. 이 앱은 Amazon Simple Email Service(Amazon SES)를 사용하여 결과와 함께 이메일 알림을 관리자에게 보냅니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Rekognition
- Amazon S3
- Amazon SES

동영상에서 사람과 객체 감지

다음 코드 예제에서는 Amazon Rekognition을 사용하여 동영상에서 사람과 객체를 감지하는 방법을 보여줍니다.

SDK for Java 2.x

Amazon Rekognition Java API를 사용하여 Amazon Simple Storage Service(Amazon S3) 버킷에 있는 동영상에서 얼굴과 객체를 감지하기 위한 앱을 만드는 방법을 보여줍니다. 이 앱은 Amazon Simple Email Service(Amazon SES)를 사용하여 결과와 함께 이메일 알림을 관리자에게 보냅니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Rekognition
- Amazon S3
- Amazon SES

- Amazon SNS
- Amazon SQS

Step Functions를 사용하여 Lambda 함수 간접 호출

다음 코드 예제에서는 AWS Lambda 함수를 순차적으로 호출하는 AWS Step Functions 상태 시스템을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

AWS Step Functions 및를 사용하여 AWS 서버리스 워크플로를 생성하는 방법을 보여줍니다 AWS SDK for Java 2.x. 각 워크플로 단계는 AWS Lambda 함수를 사용하여 구현됩니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- DynamoDB
- Lambda
- Amazon SES
- 단계 함수

Java 2.x용 SDK를 사용하는 SES API v2 예제

다음 코드 예제에서는 Amazon SES API v2와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업
- 시나리오

작업

`CreateContact`

다음 코드 예시는 `CreateContact`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
      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;
      }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateContact를 참조하세요.

`CreateContactList`

다음 코드 예시는 `CreateContactList`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateContactList를 참조하세요.

`CreateEmailIdentity`

다음 코드 예시는 `CreateEmailIdentity`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateEmailIdentity를 참조하세요.

`CreateEmailTemplate`

다음 코드 예시는 `CreateEmailTemplate`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateEmailTemplate를 참조하세요.

`DeleteContactList`

다음 코드 예시는 `DeleteContactList`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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();
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteContactList를 참조하세요.

`DeleteEmailIdentity`

다음 코드 예시는 `DeleteEmailIdentity`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        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.");
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteEmailIdentity를 참조하세요.

`DeleteEmailTemplate`

다음 코드 예시는 `DeleteEmailTemplate`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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();
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteEmailTemplate를 참조하세요.

`ListContacts`

다음 코드 예시는 `ListContacts`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListContacts를 참조하세요.

`SendEmail`

다음 코드 예시는 `SendEmail`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

메시지를 전송합니다.

```
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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * 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);
        }
    }
}
```

템플릿을 사용하여 메시지를 보냅니다.

```
      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());
     }
```

헤더 정보가 포함된 메시지를 전송합니다.

```
public class SendwithHeader {

    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();

        String bodyHTML = """
                <html>
```

```
                    <head></head>
                    <body>
                        <h1>Hello!</h1>
                        <p>See the list of customers.</p>
                    </body>
                </html>
                """;

        sendWithHeader(sesv2Client, sender, recipient, subject, bodyHTML);
        sesv2Client.close();
    }

    /**
     * Sends an email using the AWS SES V2 client.
     *
     * @param sesv2Client the SES V2 client to use for sending the email
     * @param sender the email address of the sender
     * @param recipient the email address of the recipient
     * @param subject the subject of the email
     * @param bodyHTML the HTML content of the email body
     */
    public static void sendWithHeader(SesV2Client sesv2Client,
                                      String sender,
                                      String recipient,
                                      String subject,
                                      String bodyHTML) {
        EmailContent emailContent = EmailContent.builder()
                .simple(Message.builder()
                        .body(b -> b.html(c ->
 c.charset(UTF_8.name()).data(bodyHTML))
                                .text(c -> c.charset(UTF_8.name()).data(bodyHTML)))
                        .subject(c -> c.charset(UTF_8.name()).data(subject))
                        .headers(List.of(
                                MessageHeader.builder()
                                        .name("List-Unsubscribe")
                                        .value("<https://nutrition.co/?
address=x&topic=x>, <mailto:unsubscribe@nutrition.co?subject=TopicUnsubscribe>")
                                        .build(),
                                MessageHeader.builder()
                                        .name("List-Unsubscribe-Post")
                                        .value("List-Unsubscribe=One-Click")
                                        .build()))
                        .build())
                .build();
```

```
        SendEmailRequest request = SendEmailRequest.builder()
                .fromEmailAddress(sender)
                .destination(d -> d.toAddresses(recipient))
                .content(emailContent)
                .build();

        try {
            SendEmailResponse response = sesv2Client.sendEmail(request);
            System.out.println("Email sent! Message ID: " + response.messageId());
        } catch (SesV2Exception e) {
            System.err.println("Failed to send email: " +
 e.awsErrorDetails().errorMessage());
            throw new RuntimeException(e);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 SendEmail을 참조하세요.

시나리오

뉴스레터 시나리오

다음 코드 예제에서는 Amazon SES API v2 뉴스레터 시나리오를 실행하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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();
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - CreateContact
  - CreateContactList
  - CreateEmailIdentity
  - CreateEmailTemplate
  - DeleteContactList
  - DeleteEmailIdentity
  - DeleteEmailTemplate
  - ListContacts
  - SendEmail.simple
  - SendEmail.template

Java 2.x용 SDK를 사용하는 Amazon SNS 예제

다음 코드 예제에서는 AWS SDK for Java 2.x Amazon SNS에서를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시작하기
- 작업
- 시나리오

- 서버리스 예제

시작하기

Hello Amazon SNS

다음 코드 예제에서는 Amazon SNS 사용을 시작하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListTopics를 참조하세요.

작업

`CheckIfPhoneNumberIsOptedOut`

다음 코드 예시는 `CheckIfPhoneNumberIsOptedOut`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
       }
```

```
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CheckIfPhoneNumberIsOptedOut을 참조하세요.

`ConfirmSubscription`

다음 코드 예시는 `ConfirmSubscription`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ConfirmSubscription을 참조하세요.

`CreateTopic`

다음 코드 예시는 `CreateTopic`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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 "";
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateTopic을 참조하세요.

`DeleteTopic`

다음 코드 예시는 `DeleteTopic`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteTopic을 참조하세요.

`GetSMSAttributes`

다음 코드 예시는 `GetSMSAttributes`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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");
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetSMSAttributes를 참조하세요.

`GetTopicAttributes`

다음 코드 예시는 `GetTopicAttributes`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetTopicAttributes를 참조하세요.

`ListPhoneNumbersOptedOut`

다음 코드 예시는 `ListPhoneNumbersOptedOut`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListPhoneNumbersOptedOut을 참조하세요.

`ListSubscriptions`

다음 코드 예시는 `ListSubscriptions`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListSubscriptions를 참조하세요.

`ListTopics`

다음 코드 예시는 `ListTopics`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListTopics를 참조하세요.

Publish

다음 코드 예시는 Publish의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Publish를 참조하세요.

`SetSMSAttributes`

다음 코드 예시는 `SetSMSAttributes`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
```

```
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 SetSMSAttributes를 참조하세요.

SetSubscriptionAttributes

다음 코드 예시는 SetSubscriptionAttributes의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 SetSubscriptionAttributes를 참조하세요.

`SetTopicAttributes`

다음 코드 예시는 `SetTopicAttributes`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 SetTopicAttributes를 참조하세요.

Subscribe

다음 코드 예시는 Subscribe의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

이메일 주소로 주제 구독.

```
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);
        }
    }
}
```

HTTP 엔드포인트에서 주제를 구독합니다.

```
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);
        }
    }
}
```

Lambda 함수에서 주제를 구독합니다.

```
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 "";
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Subscribe를 참조하세요.

`TagResource`

다음 코드 예시는 TagResource의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 TagResource를 참조하세요.

`Unsubscribe`

다음 코드 예시는 `Unsubscribe`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Unsubscribe를 참조하세요.

시나리오

DynamoDB 테이블에 데이터를 제출하기 위한 앱 구축

다음 코드 예제에서는 Amazon DynamoDB 테이블에 데이터를 제출하고 사용자가 테이블을 업데이트하면 알려주는 애플리케이션을 빌드하는 방법을 보여줍니다.

SDK for Java 2.x

Amazon DynamoDB Java API를 사용하여 데이터를 제출하고 Amazon Simple Notification Service Java API를 사용하여 문자 메시지를 전송하는 동적 웹 애플리케이션을 생성하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- DynamoDB
- Amazon SNS

Amazon SNS 애플리케이션 구축

다음 코드 예제에서는 구독 및 게시 기능이 있고 메시지를 번역하는 애플리케이션을 만드는 방법을 보여줍니다.

SDK for Java 2.x

Amazon Simple Notification Service Java API를 사용하여 구독 및 게시 기능이 있는 웹 애플리케이션을 생성하는 방법을 보여줍니다. 또한 이 예제 애플리케이션은 메시지를 번역합니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

Java Async API를 사용한 예제를 설정하고 실행하는 방법에 대한 전체 소스 코드와 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon SNS
- Amazon Translate

푸시 알림에 대한 플랫폼 엔드포인트 생성

다음 코드 예제에서는 Amazon SNS 푸시 알림에 대한 플랫폼 엔드포인트를 생성하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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 device token or registration ID of the mobile device.
 This is a unique
               identifier provided by the device platform (e.g., Apple Push
 Notification Service (APNS) for iOS devices, Firebase Cloud Messaging (FCM)
               for Android devices) when the mobile app is registered to receive
 push notifications.

               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);
            return;
```

```
        }

        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());
        }
    }
}
```

사진을 관리하기 위한 서버리스 애플리케이션 만들기

다음 코드 예시에서는 사용자가 레이블을 사용하여 사진을 관리할 수 있는 서버리스 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 AWS 커뮤니티의 게시물을 참조하세요.

이 예제에서 사용되는 서비스

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

FIFO 주제에 생성 및 게시

다음 코드 예제는 FIFO Amazon SNS 주제를 생성하고 거기에 게시하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예에서는

- Amazon SNS FIFO 주제 1개, Amazon SQS FIFO 대기열 2개, 표준 대기열 1개를 생성합니다.
- 대기열에서 주제를 구독하고 해당 주제에 메시지를 게시합니다.

테스트에서는 각 대기열에 대한 메시지 수신 여부를 확인합니다. 또한 전체 예제에서는 액세스 정책을 추가하는 것을 보여주고 마지막에 리소스를 삭제합니다.

```
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",
                "FifoThroughputScope", "MessageGroup");

            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);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - CreateTopic
  - 게시
  - Subscribe

동영상에서 사람과 객체 감지

다음 코드 예제에서는 Amazon Rekognition을 사용하여 동영상에서 사람과 객체를 감지하는 방법을 보여줍니다.

SDK for Java 2.x

Amazon Rekognition Java API를 사용하여 Amazon Simple Storage Service(Amazon S3) 버킷에 있는 동영상에서 얼굴과 객체를 감지하기 위한 앱을 만드는 방법을 보여줍니다. 이 앱은 Amazon Simple Email Service(Amazon SES)를 사용하여 결과와 함께 이메일 알림을 관리자에게 보냅니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

주제에 SMS 메시지 게시

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- Amazon SNS 주제를 생성합니다.
- 전화번호를 주제에 구독시킵니다.
- 모든 구독 전화번호가 한 번에 메시지를 받을 수 있도록 주제에 SMS 메시지를 게시합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

주제를 만들고 해당 ARN을 반환합니다.

```
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 "";
    }
}
```

주제에 엔드포인트를 구독 설정합니다.

```
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);
        }
    }
}
```

발신자의 ID, 최고 가격 및 유형과 같은 메시지의 속성을 설정합니다. 메시지 속성은 선택 사항입니다.

```
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);
        }
    }
}
```

주제에 메시지를 게시합니다. 메시지는 모든 구독자에게 전송됩니다.

```
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);
        }
    }
}
```

SMS 문자 메시지 게시

다음 코드 예제에서는 Amazon SNS를 사용하여 SMS 메시지를 게시하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Publish를 참조하세요.

대기열에 메시지 게시

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 주제(FIFO 또는 비 FIFO)를 생성합니다.
- 필터 적용 옵션을 사용하여 여러 개의 대기열로 주제를 구독합니다.
- 주제에 메시지를 게시합니다.
- 대기열에서 받은 메시지를 폴링합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 * <p>
 * This Java example performs these tasks:
 * <p>
 * 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 = args[0];
        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 scenario, 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 = """
        {
             "Statement": [
             {
                 "Effect": "Allow",
                         "Principal": {
```

```
                    "Service": "sns.amazonaws.com"
                },
                "Action": "sqs:SendMessage",
                        "Resource": "arn:aws:sqs:us-east-1:%s:%s",
                        "Condition": {
                    "ArnEquals": {
                        "aws:SourceArn": "arn:aws:sns:us-east-1:%s:%s"
                    }
                }
            }
            ]
        }
        """.formatted(accountId, sqsQueueName, accountId, topicName);

        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 "";
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

API Gateway를 사용하여 Lambda 함수 간접 호출

다음 코드 예제에서는 Amazon API Gateway에서 호출한 AWS Lambda 함수를 생성하는 방법을 보여 줍니다.

SDK for Java 2.x

Lambda Java 런타임 API를 사용하여 AWS Lambda 함수를 생성하는 방법을 보여줍니다. 이 예제에서는 다양한 AWS 서비스를 호출하여 특정 사용 사례를 수행합니다. 이 예제에서는 Amazon API Gateway에서 간접 호출한 Lambda 함수를 생성하여 작업 기념일에 대한 Amazon DynamoDB 테이블을 스캔하고 Amazon Simple Notification Service(Amazon SNS)를 사용하여 직원에게 1주년 기념일을 축하하는 문자 메시지를 전송하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

예약된 이벤트를 사용하여 Lambda 함수 간접 호출

다음 코드 예제에서는 Amazon EventBridge 예약 이벤트에서 호출된 AWS Lambda 함수를 생성하는 방법을 보여줍니다.

SDK for Java 2.x

AWS Lambda 함수를 호출하는 Amazon EventBridge 예약 이벤트를 생성하는 방법을 보여줍니다. Lambda 함수가 간접 호출될 때 cron 표현식을 사용하여 일정을 예약하도록 EventBridge를 구성합니다. 이 예제에서는 Lambda Java 런타임 API를 사용하여 Lambda 함수를 생성합니다. 이 예제에서는 다양한 AWS 서비스를 호출하여 특정 사용 사례를 수행합니다. 이 예제에서는 1주년 기념일에 직원에게 축하하는 모바일 문자 메시지를 전송하는 앱을 생성하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- CloudWatch Logs
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

서버리스 예제

Amazon SNS 트리거를 사용하여 Lambda 함수 간접 호출

다음 코드 예제에서는 SNS 주제의 메시지를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 함수는 이벤트 파라미터에서 메시지를 검색하고 각 메시지의 내용을 로깅합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Java를 사용하여 Lambda로 SNS 이벤트를 사용합니다.

```
// 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);
        }
    }

}
```

Java 2.x용 SDK를 사용하는 Amazon SQS 예제

다음 코드 예제에서는 Amazon SQS와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시작하기

Hello Amazon SNS

다음 코드 예제에서는 Amazon SQS 사용을 시작하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListQueues를 참조하세요.

작업

`CreateQueue`

다음 코드 예시는 `CreateQueue`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateQueue를 참조하세요.

`DeleteMessage`

다음 코드 예시는 `DeleteMessage`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        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);
        }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteMessage를 참조하세요.

`DeleteQueue`

다음 코드 예시는 `DeleteQueue`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteQueue를 참조하세요.

`GetQueueUrl`

다음 코드 예시는 `GetQueueUrl`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
            GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
            return getQueueUrlResponse.queueUrl();
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetQueueUrl을 참조하세요.

`ListQueues`

다음 코드 예시는 `ListQueues`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        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);
        }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListQueues를 참조하세요.

`ReceiveMessage`

다음 코드 예시는 `ReceiveMessage`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        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;
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ReceiveMessage를 참조하세요.

`SendMessage`

다음 코드 예시는 `SendMessage`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

`SendMessage` 작업의 두 가지 예는 다음과 같습니다.

- 지연을 적용하여 본문이 포함된 메시지 전송

- 본문 및 메시지 속성이 포함된 메시지 전송

지연을 적용하여 본문이 포함된 메시지를 전송합니다.

```
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);
        }
    }
}
```

본문 및 메시지 속성이 포함된 메시지를 전송합니다.

```
    /**
     * <p>This method demonstrates how to add message attributes to a message.
     * Each attribute must specify a name, value, and data type. You use a Java Map
 to supply the attributes. The map's
     * key is the attribute name, and you specify the map's entry value using a
 builder that includes the attribute
     * value and data type.</p>
     *
     * <p>The data type must start with one of "String", "Number" or "Binary". You
 can optionally
     * define a custom extension by using a "." and your extension.</p>
```

```
     *
     * <p>The SQS Developer Guide provides more information on @see <a
     * href="https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-message-metadata.html#sqs-message-attributes">message
     * attributes</a>.</p>
     *
     * @param thumbailPath Filesystem path of the image.
     * @param queueUrl     URL of the SQS queue.
     */
    static void sendMessageWithAttributes(Path thumbailPath, String queueUrl) {
        Map<String, MessageAttributeValue> messageAttributeMap;
        try {
            messageAttributeMap = Map.of(
                    "Name", MessageAttributeValue.builder()
                            .stringValue("Jane Doe")
                            .dataType("String").build(),
                    "Age", MessageAttributeValue.builder()
                            .stringValue("42")
                            .dataType("Number.int").build(),
                    "Image", MessageAttributeValue.builder()

 .binaryValue(SdkBytes.fromByteArray(Files.readAllBytes(thumbailPath)))
                            .dataType("Binary.jpg").build()
            );
        } catch (IOException e) {
            LOGGER.error("An I/O exception occurred reading thumbnail image: {}",
 e.getMessage(), e);
            throw new RuntimeException(e);
        }

        SendMessageRequest request = SendMessageRequest.builder()
                .queueUrl(queueUrl)
                .messageBody("Hello SQS")
                .messageAttributes(messageAttributeMap)
                .build();
        try {
            SendMessageResponse sendMessageResponse =
 SQS_CLIENT.sendMessage(request);
            LOGGER.info("Message ID: {}", sendMessageResponse.messageId());
        } catch (SqsException e) {
            LOGGER.error("Exception occurred sending message: {}", e.getMessage(),
 e);
            throw new RuntimeException(e);
        }
```

```
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 SendMessage를 참조하세요.

SendMessageBatch

다음 코드 예시는 SendMessageBatch의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
            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);
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 SendMessageBatch를 참조하세요.

SetQueueAttributes

다음 코드 예시는 SetQueueAttributes의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

사용자 지정 KMS 키를 사용하여 서버 측 암호화(SSE)를 사용하도록 Amazon SQS를 구성합니다.

```
    public static void addEncryption(String queueName, String kmsMasterKeyAlias) {
        SqsClient sqsClient = SqsClient.create();

        GetQueueUrlRequest urlRequest = GetQueueUrlRequest.builder()
                .queueName(queueName)
                .build();

        GetQueueUrlResponse getQueueUrlResponse;
        try {
            getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest);
        } catch (QueueDoesNotExistException e) {
            LOGGER.error(e.getMessage(), e);
            throw new RuntimeException(e);
        }
        String queueUrl = getQueueUrlResponse.queueUrl();


        Map<QueueAttributeName, String> attributes = Map.of(
                QueueAttributeName.KMS_MASTER_KEY_ID, kmsMasterKeyAlias,
                QueueAttributeName.KMS_DATA_KEY_REUSE_PERIOD_SECONDS, "140" // Set
the data key reuse period to 140 seconds.
        );                                                                  // This
is how long SQS can reuse the data key before requesting a new one from KMS.

        SetQueueAttributesRequest attRequest = SetQueueAttributesRequest.builder()
                .queueUrl(queueUrl)
                .attributes(attributes)
                .build();
        try {
            sqsClient.setQueueAttributes(attRequest);
            LOGGER.info("The attributes have been applied to {}", queueName);
        } catch (InvalidAttributeNameException | InvalidAttributeValueException e) {
            LOGGER.error(e.getMessage(), e);
```

```
            throw new RuntimeException(e);
        } finally {
            sqsClient.close();
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 SetQueueAttributes를 참조하세요.

시나리오

메시징 애플리케이션 생성

다음 코드 예제에서는 Amazon SQS를 사용하여 메시징 애플리케이션을 만드는 방법을 보여줍니다.

SDK for Java 2.x

Amazon SQS API를 사용하여 메시지를 보내고 검색하는 Spring REST API를 개발하는 방법을 보여 줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Comprehend
- Amazon SQS

FIFO 주제에 생성 및 게시

다음 코드 예제는 FIFO Amazon SNS 주제를 생성하고 거기에 게시하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예에서는

- Amazon SNS FIFO 주제 1개, Amazon SQS FIFO 대기열 2개, 표준 대기열 1개를 생성합니다.

- 대기열에서 주제를 구독하고 해당 주제에 메시지를 게시합니다.

테스트에서는 각 대기열에 대한 메시지 수신 여부를 확인합니다. 또한 전체 예제에서는 액세스 정책을 추가하는 것을 보여주고 마지막에 리소스를 삭제합니다.

```
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",
                "FifoThroughputScope", "MessageGroup");

            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);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - CreateTopic
  - 게시
  - Subscribe

동영상에서 사람과 객체 감지

다음 코드 예제에서는 Amazon Rekognition을 사용하여 동영상에서 사람과 객체를 감지하는 방법을 보여줍니다.

SDK for Java 2.x

Amazon Rekognition Java API를 사용하여 Amazon Simple Storage Service(Amazon S3) 버킷에 있는 동영상에서 얼굴과 객체를 감지하기 위한 앱을 만드는 방법을 보여줍니다. 이 앱은 Amazon Simple Email Service(Amazon SES)를 사용하여 결과와 함께 이메일 알림을 관리자에게 보냅니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS

- Amazon SQS

S3를 사용하여 대규모 메시지 관리

다음 코드 예제에서는 Amazon SQS 확장 클라이언트 라이브러리를 사용하여 대규모 Amazon SQS 메시지를 작업하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.joda.time.DateTime;
import org.joda.time.format.DateTimeFormat;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
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.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.LifecycleExpiration;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueResponse;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.Message;
```

```
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;

import java.util.Arrays;
import java.util.List;
import java.util.UUID;

/**
 * Example of using Amazon SQS Extended Client Library for Java 2.x.
 */
public class SqsExtendedClientExample {
    private static final Logger logger =
 LoggerFactory.getLogger(SqsExtendedClientExample.class);

    private String s3BucketName;
    private String queueUrl;
    private final String queueName;
    private final S3Client s3Client;
    private final SqsClient sqsExtendedClient;
    private final int messageSize;

    /**
     * Constructor with default clients and message size.
     */
    public SqsExtendedClientExample() {
        this(S3Client.create(), 300000);
    }

    /**
     * Constructor with custom S3 client and message size.
     *
     * @param s3Client The S3 client to use
     * @param messageSize The size of the test message to create
     */
    public SqsExtendedClientExample(S3Client s3Client, int messageSize) {
        this.s3Client = s3Client;
        this.messageSize = messageSize;

        // Generate a unique bucket name.
        this.s3BucketName = UUID.randomUUID() + "-" +
                DateTimeFormat.forPattern("yyMMdd-hhmmss").print(new DateTime());

        // Generate a unique queue name.
```

```
        this.queueName = "MyQueue-" + UUID.randomUUID();

        // Configure the SQS extended client.
        final ExtendedClientConfiguration extendedClientConfig = new
ExtendedClientConfiguration()
                .withPayloadSupportEnabled(s3Client, s3BucketName);

        this.sqsExtendedClient = new
AmazonSQSExtendedClient(SqsClient.builder().build(), extendedClientConfig);
    }

    public static void main(String[] args) {
        SqsExtendedClientExample example = new SqsExtendedClientExample();
        try {
            example.setup();
            example.sendAndReceiveMessage();
        } finally {
            example.cleanup();
        }
    }

    /**
     * Send a large message and receive it back.
     *
     * @return The received message
     */
    public Message sendAndReceiveMessage() {
        try {
            // Create a large message.
            char[] chars = new char[messageSize];
            Arrays.fill(chars, 'x');
            String largeMessage = new String(chars);

            // Send the message.
            final SendMessageRequest sendMessageRequest =
SendMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageBody(largeMessage)
                    .build();

            sqsExtendedClient.sendMessage(sendMessageRequest);
            logger.info("Sent message of size: {}", largeMessage.length());

            // Receive and return the message.
```

```
            final ReceiveMessageResponse receiveMessageResponse =
sqsExtendedClient.receiveMessage(
                    ReceiveMessageRequest.builder().queueUrl(queueUrl).build());

            List<Message> messages = receiveMessageResponse.messages();
            if (messages.isEmpty()) {
                throw new RuntimeException("No messages received");
            }

            Message message = messages.getFirst();
            logger.info("\nMessage received.");
            logger.info("  ID: {}", message.messageId());
            logger.info("  Receipt handle: {}", message.receiptHandle());
            logger.info("  Message body size: {}", message.body().length());
            logger.info("  Message body (first 5 characters): {}",
message.body().substring(0, 5));

            return message;
        } catch (RuntimeException e) {
            logger.error("Error during message processing: {}", e.getMessage(), e);
            throw e;
        }
    }
```

- 자세한 정보는 AWS SDK for Java 2.x 개발자 안내서를 참조하세요.
- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - CreateBucket
  - PutBucketLifecycleConfiguration
  - ReceiveMessage
  - SendMessage

S3 이벤트 알림 처리

다음 코드 예시에서는 객체 지향적 방식으로 S3 이벤트 알림을 사용하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예시에서는 Amazon SQS를 사용하여 S3 알림 이벤트를 처리하는 방법을 보여줍니다.

```
    /**
     * This method receives S3 event notifications by using an SqsAsyncClient.
     * After the client receives the messages it deserializes the JSON payload and
 logs them. It uses
     * the S3EventNotification class (part of the S3 event notification API for
 Java) to deserialize
     * the JSON payload and access the messages in an object-oriented way.
     *
     * @param queueUrl The URL of the AWS SQS queue that receives the S3 event
 notifications.
     * @see <a href="https://sdk.amazonaws.com/java/api/latest/software/amazon/
awssdk/eventnotifications/s3/model/package-summary.html">S3EventNotification API</
a>.
     * <p>
     * To use S3 event notification serialization/deserialization to objects, add
 the following
     * dependency to your Maven pom.xml file.
     * <dependency>
     * <groupId>software.amazon.awssdk</groupId>
     * <artifactId>s3-event-notifications</artifactId>
     * <version><LATEST></version>
     * </dependency>
     * <p>
     * The S3 event notification API became available with version 2.25.11 of the
 Java SDK.
     * <p>
     * This example shows the use of the API with AWS SQS, but it can be used to
 process S3 event notifications
     * in AWS SNS or AWS Lambda as well.
     * <p>
     * Note: The S3EventNotification class does not work with messages routed
 through AWS EventBridge.
     */
```

```
   static void processS3Events(String bucketName, String queueUrl, String queueArn)
{
       try {
           // Configure the bucket to send Object Created and Object Tagging
notifications to an existing SQS queue.
           s3Client.putBucketNotificationConfiguration(b -> b
                   .notificationConfiguration(ncb -> ncb
                           .queueConfigurations(qcb -> qcb
                                   .events(Event.S3_OBJECT_CREATED,
Event.S3_OBJECT_TAGGING)
                                   .queueArn(queueArn)))
                           .bucket(bucketName)
           ).join();

           triggerS3EventNotifications(bucketName);
           // Wait for event notifications to propagate.
           Thread.sleep(Duration.ofSeconds(5).toMillis());

           boolean didReceiveMessages = true;
           while (didReceiveMessages) {
               // Display the number of messages that are available in the queue.
               sqsClient.getQueueAttributes(b -> b
                               .queueUrl(queueUrl)

.attributeNames(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)
                       ).thenAccept(attributeResponse ->
                               logger.info("Approximate number of messages in the
queue: {}",

attributeResponse.attributes().get(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)))
                       .join();

               // Receive the messages.
               ReceiveMessageResponse response = sqsClient.receiveMessage(b -> b
                       .queueUrl(queueUrl)
               ).get();
               logger.info("Count of received messages: {}",
response.messages().size());
               didReceiveMessages = !response.messages().isEmpty();

               // Create a collection to hold the received message for deletion
               // after we log the messages.
               HashSet<DeleteMessageBatchRequestEntry> messagesToDelete = new
HashSet<>();
```

```
                // Process each message.
                response.messages().forEach(message -> {
                    logger.info("Message id: {}", message.messageId());
                    // Deserialize JSON message body to a S3EventNotification object
                    // to access messages in an object-oriented way.
                    S3EventNotification event =
S3EventNotification.fromJson(message.body());

                    // Log the S3 event notification record details.
                    if (event.getRecords() != null) {
                        event.getRecords().forEach(record -> {
                            String eventName = record.getEventName();
                            String key = record.getS3().getObject().getKey();
                            logger.info(record.toString());
                            logger.info("Event name is {} and key is {}", eventName,
key);
                        });
                    }
                    // Add logged messages to collection for batch deletion.
                    messagesToDelete.add(DeleteMessageBatchRequestEntry.builder()
                            .id(message.messageId())
                            .receiptHandle(message.receiptHandle())
                            .build());
                });
                // Delete messages.
                if (!messagesToDelete.isEmpty()) {
                    sqsClient.deleteMessageBatch(DeleteMessageBatchRequest.builder()
                            .queueUrl(queueUrl)
                            .entries(messagesToDelete)
                            .build()
                    ).join();
                }
            } // End of while block.
        } catch (InterruptedException | ExecutionException e) {
            throw new RuntimeException(e);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - DeleteMessageBatch
  - GetQueueAttributes

- PutBucketNotificationConfiguration
- ReceiveMessage

대기열에 메시지 게시

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 주제(FIFO 또는 비 FIFO)를 생성합니다.
- 필터 적용 옵션을 사용하여 여러 개의 대기열로 주제를 구독합니다.
- 주제에 메시지를 게시합니다.
- 대기열에서 받은 메시지를 폴링합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 * <p>
 * This Java example performs these tasks:
 * <p>
 * 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 = args[0];
        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 scenario, 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 = """
        {
             "Statement": [
             {
                 "Effect": "Allow",
                         "Principal": {
                     "Service": "sns.amazonaws.com"
                 },
                 "Action": "sqs:SendMessage",
                         "Resource": "arn:aws:sqs:us-east-1:%s:%s",
                         "Condition": {
                     "ArnEquals": {
                         "aws:SourceArn": "arn:aws:sns:us-east-1:%s:%s"
                     }
                 }
             }
             ]
         }
        """.formatted(accountId, sqsQueueName, accountId, topicName);

        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 "";
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

메시지 일괄 전송 및 수신

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- Amazon SQS 대기열을 생성합니다.
- 대기열에 메시지를 일괄 전송합니다.
- 대기열에서 메시지를 일괄 수신합니다.
- 대기열에서 메시지 배치를 삭제합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

다음 예제와 같이 AWS SDK for Java 2.x와 함께 두 가지 접근 방식을 사용하여 Amazon SQS에서 배치 메시지 작업을 처리할 수 있습니다.

SendRecvBatch.java는 명시적 배치 작업을 사용합니다. 메시지 배치를 수동으로 생성하고 `sendMessageBatch()` 및 `deleteMessageBatch()`를 직접적으로 호출합니다. 또한 실패한 메시지를 포함하여 배치 응답을 처리합니다. 이 접근 방식을 사용하면 배치 크기 조정 및 오류 처리를 완벽하게 제어할 수 있습니다. 그러나 배치 처리 로직을 관리하려면 더 많은 코드가 필요합니다.

SimpleProducerConsumer.java는 자동 요청 배치 처리를 위해 상위 수준 `SqsAsyncBatchManager` 라이브러리를 사용합니다. 표준 클라이언트와 동일한 메서드 서명으로 개별 `sendMessage()` 및 `deleteMessage()` 직접 호출을 수행합니다. SDK는 이러한 직접 호출을 자동으로 버퍼링하고 배치 작업 형태로 전송합니다. 이 접근 방식을 사용하려면 코드 변경을 최소화해야 하지만 배치 처리 성능에 유리합니다.

배치 구성 및 오류 처리를 세밀하게 제어해야 하는 경우 명시적 배치 처리를 사용합니다. 코드 변경을 최소화하면서 성능을 최적화하려면 자동 배치 처리를 사용합니다.

SendRecvBatch.java - 명시적 배치 작업을 메시지 단위로 수행합니다.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.BatchResultErrorEntry;
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.DeleteMessageBatchResponse;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchResultEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.MessageAttributeValue;
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.SendMessageBatchResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchResultEntry;
import software.amazon.awssdk.services.sqs.model.SqsException;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
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
 */

/**
 * This code demonstrates basic message operations in Amazon Simple Queue Service
 (Amazon SQS).
 */

public class SendRecvBatch {
    private static final Logger LOGGER =
 LoggerFactory.getLogger(SendRecvBatch.class);
    private static final SqsClient sqsClient = SqsClient.create();


    public static void main(String[] args) {
        usageDemo();
    }
    /**
     * Send a batch of messages in a single request to an SQS queue.
     * This request may return overall success even when some messages were not
 sent.
     * The caller must inspect the Successful and Failed lists in the response and
     * resend any failed messages.
     *
```

```
    * @param queueUrl  The URL of the queue to receive the messages.
    * @param messages  The messages to send to the queue. Each message contains a
body and attributes.
    * @return The response from SQS that contains the list of successful and failed
messages.
    */
   public static SendMessageBatchResponse sendMessages(
           String queueUrl, List<MessageEntry> messages) {

       try {
           List<SendMessageBatchRequestEntry> entries = new ArrayList<>();

           for (int i = 0; i < messages.size(); i++) {
               MessageEntry message = messages.get(i);
               entries.add(SendMessageBatchRequestEntry.builder()
                       .id(String.valueOf(i))
                       .messageBody(message.getBody())
                       .messageAttributes(message.getAttributes())
                       .build());
           }

           SendMessageBatchRequest sendBatchRequest =
SendMessageBatchRequest.builder()
                   .queueUrl(queueUrl)
                   .entries(entries)
                   .build();

           SendMessageBatchResponse response =
sqsClient.sendMessageBatch(sendBatchRequest);

           if (!response.successful().isEmpty()) {
               for (SendMessageBatchResultEntry resultEntry :
response.successful()) {
                   LOGGER.info("Message sent: {}: {}", resultEntry.messageId(),

messages.get(Integer.parseInt(resultEntry.id())).getBody());
               }
           }

           if (!response.failed().isEmpty()) {
               for (BatchResultErrorEntry errorEntry : response.failed()) {
                   LOGGER.warn("Failed to send: {}: {}", errorEntry.id(),

messages.get(Integer.parseInt(errorEntry.id())).getBody());
```

```
                }
            }

            return response;

        } catch (SqsException e) {
            LOGGER.error("Send messages failed to queue: {}", queueUrl, e);
            throw e;
        }
    }

    /**
     * Receive a batch of messages in a single request from an SQS queue.
     *
     * @param queueUrl   The URL of the queue from which to receive messages.
     * @param maxNumber  The maximum number of messages to receive (capped at 10 by
SQS).
     *                   The actual number of messages received might be less.
     * @param waitTime   The maximum time to wait (in seconds) before returning.
When
     *                   this number is greater than zero, long polling is used.
This
     *                   can result in reduced costs and fewer false empty
responses.
     * @return The list of Message objects received. These each contain the body
     *         of the message and metadata and custom attributes.
     */
    public static List<Message> receiveMessages(String queueUrl, int maxNumber, int
waitTime) {
        try {
            ReceiveMessageRequest receiveRequest = ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .maxNumberOfMessages(maxNumber)
                    .waitTimeSeconds(waitTime)
                    .messageAttributeNames("All")
                    .build();

            List<Message> messages =
sqsClient.receiveMessage(receiveRequest).messages();

            for (Message message : messages) {
                LOGGER.info("Received message: {}: {}", message.messageId(),
message.body());
            }
```

```
            return messages;

        } catch (SqsException e) {
            LOGGER.error("Couldn't receive messages from queue: {}", queueUrl, e);
            throw e;
        }
    }

    /**
     * Delete a batch of messages from a queue in a single request.
     *
     * @param queueUrl  The URL of the queue from which to delete the messages.
     * @param messages  The list of messages to delete.
     * @return The response from SQS that contains the list of successful and failed
     *         message deletions.
     */
    public static DeleteMessageBatchResponse deleteMessages(String queueUrl,
List<Message> messages) {
        try {
            List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();

            for (int i = 0; i < messages.size(); i++) {
                entries.add(DeleteMessageBatchRequestEntry.builder()
                        .id(String.valueOf(i))
                        .receiptHandle(messages.get(i).receiptHandle())
                        .build());
            }

            DeleteMessageBatchRequest deleteRequest =
DeleteMessageBatchRequest.builder()
                    .queueUrl(queueUrl)
                    .entries(entries)
                    .build();

            DeleteMessageBatchResponse response =
sqsClient.deleteMessageBatch(deleteRequest);

            if (!response.successful().isEmpty()) {
                for (DeleteMessageBatchResultEntry resultEntry :
response.successful()) {
                    LOGGER.info("Deleted {}",
messages.get(Integer.parseInt(resultEntry.id())).receiptHandle());
                }
```

```
            }

            if (!response.failed().isEmpty()) {
                for (BatchResultErrorEntry errorEntry : response.failed()) {
                    LOGGER.warn("Could not delete {}",
messages.get(Integer.parseInt(errorEntry.id())).receiptHandle());
                }
            }

            return response;

        } catch (SqsException e) {
            LOGGER.error("Couldn't delete messages from queue {}", queueUrl, e);
            throw e;
        }
    }

    /**
     * Helper class to represent a message with body and attributes.
     */
    public static class MessageEntry {
        private final String body;
        private final Map<String, MessageAttributeValue> attributes;

        public MessageEntry(String body, Map<String, MessageAttributeValue>
attributes) {
            this.body = body;
            this.attributes = attributes != null ? attributes : new HashMap<>();
        }

        public String getBody() {
            return body;
        }

        public Map<String, MessageAttributeValue> getAttributes() {
            return attributes;
        }
    }

    /**
     * Shows how to:
     * * Read the lines from a file and send the lines in
     *   batches of 10 as messages to a queue.
     * * Receive the messages in batches until the queue is empty.
```

```
    * * Reassemble the lines of the file and verify they match the original file.
    */
   public static void usageDemo() {
       LOGGER.info("-".repeat(88));
       LOGGER.info("Welcome to the Amazon Simple Queue Service (Amazon SQS)
demo!");
       LOGGER.info("-".repeat(88));

       String queueUrl = null;
       try {
           // Create a queue for the demo.
           String queueName = "sqs-usage-demo-message-wrapper-" +
System.currentTimeMillis();
           CreateQueueRequest createRequest = CreateQueueRequest.builder()
                   .queueName(queueName)
                   .build();
           queueUrl = sqsClient.createQueue(createRequest).queueUrl();
           LOGGER.info("Created queue: {}", queueUrl);

           try (InputStream inputStream =
SendRecvBatch.class.getResourceAsStream("/log4j2.xml");
                BufferedReader reader = new BufferedReader(new
InputStreamReader(inputStream))) {

               List<String> lines = reader.lines().toList();

               // Send file lines in batches.
               int batchSize = 10;
               LOGGER.info("Sending file lines in batches of {} as messages.",
batchSize);

               for (int i = 0; i < lines.size(); i += batchSize) {
                   List<MessageEntry> messageBatch = new ArrayList<>();

                   for (int j = i; j < Math.min(i + batchSize, lines.size()); j++)
{
                       String line = lines.get(j);
                       if (line == null || line.trim().isEmpty()) {
                           continue; // Skip empty lines.
                       }

                       Map<String, MessageAttributeValue> attributes = new
HashMap<>();
                       attributes.put("line", MessageAttributeValue.builder()
```

```
                                .dataType("String")
                                .stringValue(String.valueOf(j))
                                .build());

                        messageBatch.add(new MessageEntry(lines.get(j),
attributes));
                    }

                    sendMessages(queueUrl, messageBatch);
                    System.out.print(".");
                    System.out.flush();
                }

                LOGGER.info("\nDone. Sent {} messages.", lines.size());

                // Receive and process messages.
                LOGGER.info("Receiving, handling, and deleting messages in batches
of {}.", batchSize);
                String[] receivedLines = new String[lines.size()];
                boolean moreMessages = true;

                while (moreMessages) {
                    List<Message> receivedMessages = receiveMessages(queueUrl,
batchSize, 5);

                    for (Message message : receivedMessages) {
                        int lineNumber =
Integer.parseInt(message.messageAttributes().get("line").stringValue());
                        receivedLines[lineNumber] = message.body();
                    }

                    if (!receivedMessages.isEmpty()) {
                        deleteMessages(queueUrl, receivedMessages);
                    } else {
                        moreMessages = false;
                    }
                }

                LOGGER.info("\nDone.");

                // Verify that all lines were received correctly.
                boolean allLinesMatch = true;
                for (int i = 0; i < lines.size(); i++) {
                    String originalLine = lines.get(i);
```

```
                    String receivedLine = receivedLines[i] == null ? "" :
receivedLines[i];

                    if (!originalLine.equals(receivedLine)) {
                        allLinesMatch = false;
                        break;
                    }
                }

                if (allLinesMatch) {
                    LOGGER.info("Successfully reassembled all file lines!");
                } else {
                    LOGGER.info("Uh oh, some lines were missed!");
                }
            }
        } catch (SqsException e) {
            LOGGER.error("SQS operation failed", e);
        } catch (RuntimeException | IOException e) {
            LOGGER.error("Unexpected runtime error during demo", e);
        } finally {
            // Clean up by deleting the queue if it was created.
            if (queueUrl != null) {
                try {
                    DeleteQueueRequest deleteQueueRequest =
DeleteQueueRequest.builder()
                            .queueUrl(queueUrl)
                            .build();
                    sqsClient.deleteQueue(deleteQueueRequest);
                    LOGGER.info("Deleted queue: {}", queueUrl);
                } catch (SqsException e) {
                    LOGGER.error("Failed to delete queue: {}", queueUrl, e);
                }
            }
        }

        LOGGER.info("Thanks for watching!");
        LOGGER.info("-".repeat(88));
    }
}
```

SimpleProducerConsumer.java - 메시지를 자동 배치 처리합니다.

```
package com.example.sqs;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.sqs.SqsAsyncClient;
import software.amazon.awssdk.services.sqs.batchmanager.SqsAsyncBatchManager;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageResponse;
import software.amazon.awssdk.core.exception.SdkException;

import java.math.BigInteger;
import java.util.List;
import java.util.Random;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.atomic.AtomicBoolean;
import java.util.concurrent.atomic.AtomicInteger;

/**
 * Demonstrates the AWS SDK for Java 2.x Automatic Request Batching API for Amazon
 SQS.
 *
 * This example showcases the high-level SqsAsyncBatchManager library that provides
 * efficient batching and buffering for SQS operations. The batch manager offers
 * methods that directly mirror SqsAsyncClient methods—sendMessage,
 changeMessageVisibility,
 * deleteMessage, and receiveMessage—making it a drop-in replacement with minimal
 code changes.
 *
 * Key features of the SqsAsyncBatchManager:
 * - Automatic batching: The SDK automatically buffers individual requests and sends
 them
 *   as batches when maxBatchSize (default: 10) or sendRequestFrequency (default:
 200ms)
 *   thresholds are reached
```

```
 * - Familiar API: Method signatures match SqsAsyncClient exactly, requiring no
 learning curve
 * - Background optimization: The batch manager maintains internal buffers and
 handles
 *   batching logic transparently
 * - Asynchronous operations: All methods return CompletableFuture for non-blocking
 execution
 *
 * Performance benefits demonstrated:
 * - Reduced API calls: Multiple individual requests are consolidated into single
 batch operations
 * - Lower costs: Fewer API calls result in reduced SQS charges
 * - Higher throughput: Batch operations process more messages per second
 * - Efficient resource utilization: Fewer network round trips and better connection
 reuse
 *
 * This example compares:
 * 1. Single-message operations using SqsAsyncClient directly
 * 2. Batch operations using SqsAsyncBatchManager with identical method calls
 *
 * Usage patterns:
 * - Set batch size to 1 to use SqsAsyncClient for baseline performance measurement
 * - Set batch size > 1 to use SqsAsyncBatchManager for optimized batch processing
 * - Monitor real-time throughput metrics to observe performance improvements
 *
 * Prerequisites:
 * - AWS SDK for Java 2.x version 2.28.0 or later
 * - An existing SQS queue
 * - Valid AWS credentials configured
 *
 * The program displays real-time metrics showing the dramatic performance
 difference
 * between individual operations and automatic batching.
 */
public class SimpleProducerConsumer {

    // The maximum runtime of the program.
    private final static int MAX_RUNTIME_MINUTES = 60;
    private final static Logger log =
 LoggerFactory.getLogger(SimpleProducerConsumer.class);

    /**
     * Runs the SQS batching demonstration with user-configured parameters.
     *
```

```
     * Prompts for queue name, thread counts, batch size, message size, and runtime.
     * Creates producer and consumer threads to demonstrate batching performance.
     *
     * @param args command line arguments (not used)
     * @throws InterruptedException if thread operations are interrupted
     */
    public static void main(String[] args) throws InterruptedException {

        final Scanner input = new Scanner(System.in);

        System.out.print("Enter the queue name: ");
        final String queueName = input.nextLine();

        System.out.print("Enter the number of producers: ");
        final int producerCount = input.nextInt();

        System.out.print("Enter the number of consumers: ");
        final int consumerCount = input.nextInt();

        System.out.print("Enter the number of messages per batch: ");
        final int batchSize = input.nextInt();

        System.out.print("Enter the message size in bytes: ");
        final int messageSizeByte = input.nextInt();

        System.out.print("Enter the run time in minutes: ");
        final int runTimeMinutes = input.nextInt();

        // Create SQS async client and batch manager for all operations.
        // The SqsAsyncBatchManager is created from the SqsAsyncClient using the
        // batchManager() factory method, which provides default batching
configuration.
        // This high-level library automatically handles request buffering and
batching
        // while maintaining the same method signatures as SqsAsyncClient.
        final SqsAsyncClient sqsAsyncClient = SqsAsyncClient.create();
        final SqsAsyncBatchManager batchManager = sqsAsyncClient.batchManager();

        final String queueUrl =
sqsAsyncClient.getQueueUrl(GetQueueUrlRequest.builder()
                .queueName(queueName)
                .build()).join().queueUrl();

        // The flag used to stop producer, consumer, and monitor threads.
```

```
        final AtomicBoolean stop = new AtomicBoolean(false);

        // Start the producers.
        final AtomicInteger producedCount = new AtomicInteger();
        final Thread[] producers = new Thread[producerCount];
        for (int i = 0; i < producerCount; i++) {
            if (batchSize == 1) {
                producers[i] = new Producer(sqsAsyncClient, queueUrl,
messageSizeByte,
                        producedCount, stop);
            } else {
                producers[i] = new BatchProducer(batchManager, queueUrl, batchSize,
                        messageSizeByte, producedCount, stop);
            }
            producers[i].start();
        }

        // Start the consumers.
        final AtomicInteger consumedCount = new AtomicInteger();
        final Thread[] consumers = new Thread[consumerCount];
        for (int i = 0; i < consumerCount; i++) {
            if (batchSize == 1) {
                consumers[i] = new Consumer(sqsAsyncClient, queueUrl, consumedCount,
stop);
            } else {
                consumers[i] = new BatchConsumer(batchManager, queueUrl, batchSize,
                        consumedCount, stop);
            }
            consumers[i].start();
        }

        // Start the monitor thread.
        final Thread monitor = new Monitor(producedCount, consumedCount, stop);
        monitor.start();

        // Wait for the specified amount of time then stop.
        Thread.sleep(TimeUnit.MINUTES.toMillis(Math.min(runTimeMinutes,
                MAX_RUNTIME_MINUTES)));
        stop.set(true);

        // Join all threads.
        for (int i = 0; i < producerCount; i++) {
            producers[i].join();
        }
```

```
        for (int i = 0; i < consumerCount; i++) {
            consumers[i].join();
        }

        monitor.interrupt();
        monitor.join();

        // Close resources
        batchManager.close();
        sqsAsyncClient.close();
    }

    /**
     * Creates a random string of approximately the specified size in bytes.
     *
     * @param sizeByte the target size in bytes for the generated string
     * @return a random string encoded in base-32
     */
    private static String makeRandomString(int sizeByte) {
        final byte[] bs = new byte[(int) Math.ceil(sizeByte * 5 / 8)];
        new Random().nextBytes(bs);
        bs[0] = (byte) ((bs[0] | 64) & 127);
        return new BigInteger(bs).toString(32);
    }

    /**
     * Sends messages individually using SqsAsyncClient for baseline performance
measurement.
     *
     * This producer demonstrates traditional single-message operations without
batching.
     * Each sendMessage() call results in a separate API request to SQS, providing
     * a performance baseline for comparison with the batch operations.
     *
     * The sendMessage() method signature is identical to
SqsAsyncBatchManager.sendMessage(),
     * showing how the high-level batching library maintains API compatibility while
     * adding automatic optimization behind the scenes.
     */
    private static class Producer extends Thread {
        final SqsAsyncClient sqsAsyncClient;
        final String queueUrl;
        final AtomicInteger producedCount;
```

```
        final AtomicBoolean stop;
        final String theMessage;

        /**
         * Creates a producer thread for single-message operations.
         *
         * @param sqsAsyncClient the SQS client for sending messages
         * @param queueUrl the URL of the target queue
         * @param messageSizeByte the size of messages to generate
         * @param producedCount shared counter for tracking sent messages
         * @param stop shared flag to signal thread termination
         */
        Producer(SqsAsyncClient sqsAsyncClient, String queueUrl, int
messageSizeByte,
                 AtomicInteger producedCount, AtomicBoolean stop) {
            this.sqsAsyncClient = sqsAsyncClient;
            this.queueUrl = queueUrl;
            this.producedCount = producedCount;
            this.stop = stop;
            this.theMessage = makeRandomString(messageSizeByte);
        }

        /**
         * Continuously sends messages until the stop flag is set.
         *
         * Uses SqsAsyncClient.sendMessage() directly, resulting in one API call per
message.
         * This approach provides baseline performance metrics for comparison with
batching.
         * Each call blocks until the individual message is sent, demonstrating
traditional
         * one-request-per-operation behavior.
         */
        public void run() {
            try {
                while (!stop.get()) {
                    sqsAsyncClient.sendMessage(SendMessageRequest.builder()
                            .queueUrl(queueUrl)
                            .messageBody(theMessage)
                            .build()).join();
                    producedCount.incrementAndGet();
                }
            } catch (SdkException | java.util.concurrent.CompletionException e) {
```

```
                // Handle both SdkException and CompletionException from async
operations.
                // If this unlikely condition occurs, stop.
                log.error("Producer: " + e.getMessage());
                System.exit(1);
            }
        }
    }

    /**
     * Sends messages using SqsAsyncBatchManager for automatic request batching and
optimization.
     *
     * This producer demonstrates the AWS SDK for Java 2.x high-level batching
library.
     * The SqsAsyncBatchManager automatically buffers individual sendMessage() calls
and
     * sends them as batches when thresholds are reached:
     * - maxBatchSize: Maximum 10 messages per batch (default)
     * - sendRequestFrequency: 200ms timeout before sending partial batches
(default)
     *
     * Key advantages of the batching approach:
     * - Identical API: batchManager.sendMessage() has the same signature as
sqsAsyncClient.sendMessage()
     * - Automatic optimization: No code changes needed to benefit from batching
     * - Transparent buffering: The SDK handles batching logic internally
     * - Reduced API calls: Multiple messages sent in single batch requests
     * - Lower costs: Fewer API calls result in reduced SQS charges
     * - Higher throughput: Batch operations process significantly more messages per
second
     */
    private static class BatchProducer extends Thread {
        final SqsAsyncBatchManager batchManager;
        final String queueUrl;
        final int batchSize;
        final AtomicInteger producedCount;
        final AtomicBoolean stop;
        final String theMessage;

        /**
         * Creates a producer thread for batch operations.
         *
         * @param batchManager the batch manager for efficient message sending
```

```
         * @param queueUrl the URL of the target queue
         * @param batchSize the number of messages to send per batch
         * @param messageSizeByte the size of messages to generate
         * @param producedCount shared counter for tracking sent messages
         * @param stop shared flag to signal thread termination
         */
        BatchProducer(SqsAsyncBatchManager batchManager, String queueUrl, int
batchSize,
                      int messageSizeByte, AtomicInteger producedCount,
                      AtomicBoolean stop) {
            this.batchManager = batchManager;
            this.queueUrl = queueUrl;
            this.batchSize = batchSize;
            this.producedCount = producedCount;
            this.stop = stop;
            this.theMessage = makeRandomString(messageSizeByte);
        }

        /**
         * Continuously sends batches of messages using the high-level batching
library.
         *
         * Notice how batchManager.sendMessage() uses the exact same method
signature
         * and request builder pattern as SqsAsyncClient.sendMessage(). This
demonstrates
         * the drop-in replacement capability of the SqsAsyncBatchManager.
         *
         * The SDK automatically:
         * - Buffers individual sendMessage() calls internally
         * - Groups them into batch requests when thresholds are met
         * - Sends SendMessageBatchRequest operations to SQS
         * - Returns individual CompletableFuture responses for each message
         *
         * This transparent batching provides significant performance improvements
         * without requiring changes to application logic or error handling
patterns.
         */
        public void run() {
            try {
                while (!stop.get()) {
                    // Send multiple messages using the high-level batch manager.
                    // Each batchManager.sendMessage() call uses identical syntax to
```

```
                    // sqsAsyncClient.sendMessage(), demonstrating API
compatibility.
                    // The SDK automatically buffers these calls and sends them as
                    // batch operations when maxBatchSize (10) or
sendRequestFrequency (200ms)
                    // thresholds are reached, significantly improving throughput.
                    for (int i = 0; i < batchSize; i++) {
                        CompletableFuture<SendMessageResponse> future =
batchManager.sendMessage(
                                SendMessageRequest.builder()
                                        .queueUrl(queueUrl)
                                        .messageBody(theMessage)
                                        .build());

                        // Handle the response asynchronously
                        future.whenComplete((response, throwable) -> {
                            if (throwable == null) {
                                producedCount.incrementAndGet();
                            } else if (!(throwable instanceof
java.util.concurrent.CancellationException) &&
                                    !(throwable.getMessage() != null &&
throwable.getMessage().contains("executor not accepting a task"))) {
                                log.error("BatchProducer: Failed to send message",
throwable);
                            }
                            // Ignore CancellationException and executor shutdown
errors - expected during shutdown
                        });
                    }

                    // Small delay to allow batching to occur
                    Thread.sleep(10);
                }
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                log.error("BatchProducer interrupted: " + e.getMessage());
            } catch (SdkException | java.util.concurrent.CompletionException e) {
                log.error("BatchProducer: " + e.getMessage());
                System.exit(1);
            }
        }
    }

    /**
```

```
    * Receives and deletes messages individually using SqsAsyncClient for baseline
measurement.
    *
    * This consumer demonstrates traditional single-message operations without
batching.
    * Each receiveMessage() and deleteMessage() call results in separate API
requests,
    * providing a performance baseline for comparison with batch operations.
    *
    * The method signatures are identical to SqsAsyncBatchManager methods:
    * - receiveMessage() matches batchManager.receiveMessage()
    * - deleteMessage() matches batchManager.deleteMessage()
    *
    * This API consistency allows easy migration to the high-level batching
library.
    */
   private static class Consumer extends Thread {
       final SqsAsyncClient sqsAsyncClient;
       final String queueUrl;
       final AtomicInteger consumedCount;
       final AtomicBoolean stop;

       /**
        * Creates a consumer thread for single-message operations.
        *
        * @param sqsAsyncClient the SQS client for receiving messages
        * @param queueUrl the URL of the source queue
        * @param consumedCount shared counter for tracking processed messages
        * @param stop shared flag to signal thread termination
        */
       Consumer(SqsAsyncClient sqsAsyncClient, String queueUrl, AtomicInteger
consumedCount,
                AtomicBoolean stop) {
           this.sqsAsyncClient = sqsAsyncClient;
           this.queueUrl = queueUrl;
           this.consumedCount = consumedCount;
           this.stop = stop;
       }

       /**
        * Continuously receives and deletes messages using traditional single-
request operations.
        *
        * Uses SqsAsyncClient methods directly:
```

```
         * - receiveMessage(): One API call per receive operation
         * - deleteMessage(): One API call per delete operation
         *
         * This approach demonstrates the baseline performance without batching
optimization.
         * Compare these method calls with the identical signatures used in
BatchConsumer
         * to see how the high-level batching library maintains API compatibility.
         */
        public void run() {
            try {
                while (!stop.get()) {
                    try {
                        final ReceiveMessageResponse result =
sqsAsyncClient.receiveMessage(
                                ReceiveMessageRequest.builder()
                                        .queueUrl(queueUrl)
                                        .build()).join();

                        if (!result.messages().isEmpty()) {
                            final Message m = result.messages().get(0);
                            // Note: deleteMessage() signature identical to
batchManager.deleteMessage()

sqsAsyncClient.deleteMessage(DeleteMessageRequest.builder()
                                    .queueUrl(queueUrl)
                                    .receiptHandle(m.receiptHandle())
                                    .build()).join();
                            consumedCount.incrementAndGet();
                        }
                    } catch (SdkException | java.util.concurrent.CompletionException
e) {
                        log.error(e.getMessage());
                    }
                }
            } catch (SdkException | java.util.concurrent.CompletionException e) {
                // Handle both SdkException and CompletionException from async
operations.
                // If this unlikely condition occurs, stop.
                log.error("Consumer: " + e.getMessage());
                System.exit(1);
            }
        }
    }
```

```
   /**
    * Receives and deletes messages using SqsAsyncBatchManager for automatic
optimization.
    *
    * This consumer demonstrates the AWS SDK for Java 2.x high-level batching
library
    * for message consumption. The SqsAsyncBatchManager provides two key
optimizations:
    *
    * 1. Receive optimization: Maintains an internal buffer of messages fetched in
the
    *    background, so receiveMessage() calls return immediately from the buffer
    * 2. Delete batching: Automatically buffers deleteMessage() calls and sends
them
    *    as DeleteMessageBatchRequest operations when thresholds are reached
    *
    * Key features:
    * - Identical API: receiveMessage() and deleteMessage() have the same
signatures
    *   as SqsAsyncClient methods, making this a true drop-in replacement
    * - Background fetching: The batch manager continuously fetches messages to
keep
    *   the internal buffer populated, reducing receive latency
    * - Automatic delete batching: Individual deleteMessage() calls are buffered
and
    *   sent as batch operations (up to 10 per batch, 200ms frequency)
    * - Transparent optimization: No application logic changes needed to benefit
    *
    * Performance benefits:
    * - Reduced API calls through automatic batching of delete operations
    * - Lower latency for receives due to background message buffering
    * - Higher overall throughput with fewer network round trips
    */
   private static class BatchConsumer extends Thread {
       final SqsAsyncBatchManager batchManager;
       final String queueUrl;
       final int batchSize;
       final AtomicInteger consumedCount;
       final AtomicBoolean stop;

       /**
        * Creates a consumer thread for batch operations.
        *
```

```
         * @param batchManager the batch manager for efficient message processing
         * @param queueUrl the URL of the source queue
         * @param batchSize the maximum number of messages to receive per batch
         * @param consumedCount shared counter for tracking processed messages
         * @param stop shared flag to signal thread termination
         */
        BatchConsumer(SqsAsyncBatchManager batchManager, String queueUrl, int
batchSize,
                      AtomicInteger consumedCount, AtomicBoolean stop) {
            this.batchManager = batchManager;
            this.queueUrl = queueUrl;
            this.batchSize = batchSize;
            this.consumedCount = consumedCount;
            this.stop = stop;
        }

        /**
         * Continuously receives and deletes messages using the high-level batching
library.
         *
         * Demonstrates the key advantage of SqsAsyncBatchManager: identical method
signatures
         * with automatic optimization. Notice how:
         *
         * - batchManager.receiveMessage() uses the same syntax as
sqsAsyncClient.receiveMessage()
         * - batchManager.deleteMessage() uses the same syntax as
sqsAsyncClient.deleteMessage()
         *
         * Behind the scenes, the batch manager:
         * 1. Maintains an internal message buffer populated by background fetching
         * 2. Returns messages immediately from the buffer (reduced latency)
         * 3. Automatically batches deleteMessage() calls into
DeleteMessageBatchRequest operations
         * 4. Sends batch deletes when maxBatchSize (10) or sendRequestFrequency
(200ms) is reached
         *
         * This provides significant performance improvements with zero code changes
         * compared to traditional SqsAsyncClient usage patterns.
         */
        public void run() {
            try {
                while (!stop.get()) {
                    // Receive messages using the high-level batch manager.
```

```
                    // This call uses identical syntax to
sqsAsyncClient.receiveMessage()
                    // but benefits from internal message buffering for improved
performance.
                    final ReceiveMessageResponse result =
batchManager.receiveMessage(
                            ReceiveMessageRequest.builder()
                                    .queueUrl(queueUrl)
                                    .maxNumberOfMessages(Math.min(batchSize, 10))
                                    .build()).join();

                    if (!result.messages().isEmpty()) {
                        final List<Message> messages = result.messages();

                        // Delete messages using the batch manager.
                        // Each deleteMessage() call uses identical syntax to
SqsAsyncClient
                        // but the SDK automatically buffers these calls and sends
them
                        // as DeleteMessageBatchRequest operations for optimal
performance.
                        for (Message message : messages) {
                            CompletableFuture<DeleteMessageResponse> future =
batchManager.deleteMessage(
                                    DeleteMessageRequest.builder()
                                            .queueUrl(queueUrl)
                                            .receiptHandle(message.receiptHandle())
                                            .build());

                            future.whenComplete((response, throwable) -> {
                                if (throwable == null) {
                                    consumedCount.incrementAndGet();
                                } else if (!(throwable instanceof
java.util.concurrent.CancellationException) &&
                                        !(throwable.getMessage() != null &&
throwable.getMessage().contains("executor not accepting a task"))) {
                                    log.error("BatchConsumer: Failed to delete
message", throwable);
                                }
                                // Ignore CancellationException and executor
shutdown errors - expected during shutdown
                            });
                        }
                    }
```

```
                    // Small delay to prevent tight polling
                    Thread.sleep(10);
                }
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                log.error("BatchConsumer interrupted: " + e.getMessage());
            } catch (SdkException | java.util.concurrent.CompletionException e) {
                // Handle both SdkException and CompletionException from async
operations.
                // If this unlikely condition occurs, stop.
                log.error("BatchConsumer: " + e.getMessage());
                System.exit(1);
            }
        }
    }

    /**
     * Displays real-time throughput statistics every second.
     *
     * This thread logs the current count of produced and consumed messages
     * to help you monitor the performance comparison.
     */
    private static class Monitor extends Thread {
        private final AtomicInteger producedCount;
        private final AtomicInteger consumedCount;
        private final AtomicBoolean stop;

        /**
         * Creates a monitoring thread that displays throughput statistics.
         *
         * @param producedCount shared counter for messages sent
         * @param consumedCount shared counter for messages processed
         * @param stop shared flag to signal thread termination
         */
        Monitor(AtomicInteger producedCount, AtomicInteger consumedCount,
                AtomicBoolean stop) {
            this.producedCount = producedCount;
            this.consumedCount = consumedCount;
            this.stop = stop;
        }

        /**
         * Logs throughput statistics every second until stopped.
```

```
         *
         * Displays the current count of produced and consumed messages
         * to help monitor the performance comparison between batching strategies.
         */
        public void run() {
            try {
                while (!stop.get()) {
                    Thread.sleep(1000);
                    log.info("produced messages = " + producedCount.get()
                            + ", consumed messages = " + consumedCount.get());
                }
            } catch (InterruptedException e) {
                // Allow the thread to exit.
            }
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - CreateQueue
  - DeleteMessage
  - DeleteMessageBatch
  - DeleteQueue
  - ReceiveMessage
  - SendMessage
  - SendMessageBatch

Amazon SQS Java 메시징 라이브러리를 사용하여 JMS 인터페이스 작업

다음 코드 예제에서는 Amazon SQS Java 메시징 라이브러리를 사용하여 JMS 인터페이스로 작업하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

다음 예제는 표준 Amazon SQS 대기열과 함께 작동하며 다음을 포함합니다.

- 테스트 메시지를 전송합니다.
- 메시지를 동기식으로 수신합니다.
- 메시지를 비동기식으로 수신합니다.
- CLIENT_ACKNOWLEDGE 모드를 사용하여 메시지를 수신합니다.
- UNORDERED_ACKNOWLEDGE 모드를 사용하여 메시지를 수신합니다.
- Spring을 사용하여 종속성을 주입합니다.
- 다른 예제에서 사용하는 일반적인 메서드를 제공하는 유틸리티 클래스입니다.

Amazon SQS와 함께 JMS를 사용하는 방법에 대한 자세한 내용은 Amazon SQS 개발자 가이드를 참조하세요.

테스트 메시지를 전송합니다.

```
   /**
    * This method establishes a connection to a standard Amazon SQS queue using the
Amazon SQS
    * Java Messaging Library and sends text messages to it. It uses JMS (Java
Message Service) API
    * with automatic acknowledgment mode to ensure reliable message delivery, and
automatically
    * manages all messaging resources.
    *
    * @throws JMSException If there is a problem connecting to or sending messages
to the queue
    */
   public static void doSendTextMessage() throws JMSException {
       // Create a connection factory.
       SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
               new ProviderConfiguration(),
               SqsClient.create()
       );
```

```
        // Create the connection in a try-with-resources statement so that it's
closed automatically.
        try (SQSConnection connection = connectionFactory.createConnection()) {

            // Create the queue if needed.
            SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);

            // Create a session that uses the JMS auto-acknowledge mode.
            Session session = connection.createSession(false,
Session.AUTO_ACKNOWLEDGE);
            MessageProducer producer =
session.createProducer(session.createQueue(QUEUE_NAME));

            createAndSendMessages(session, producer);
        } // The connection closes automatically. This also closes the session.
        LOGGER.info("Connection closed");
    }

    /**
     * This method reads text input from the keyboard and sends each line as a
separate message
     * to a standard Amazon SQS queue using the Amazon SQS Java Messaging Library.
It continues
     * to accept input until the user enters an empty line, using JMS (Java Message
Service) API to
     * handle the message delivery.
     *
     * @param session The JMS session used to create messages
     * @param producer The JMS message producer used to send messages to the queue
     */
    private static void createAndSendMessages(Session session, MessageProducer
producer) {
        BufferedReader inputReader = new BufferedReader(
                new InputStreamReader(System.in, Charset.defaultCharset()));

        try {
            String input;
            while (true) {
                LOGGER.info("Enter message to send (leave empty to exit): ");
                input = inputReader.readLine();
                if (input == null || input.isEmpty()) break;
```

```
                TextMessage message = session.createTextMessage(input);
                producer.send(message);
                LOGGER.info("Send message {}", message.getJMSMessageID());
            }
        } catch (EOFException e) {
            // Just return on EOF
        } catch (IOException e) {
            LOGGER.error("Failed reading input: {}", e.getMessage(), e);
        } catch (JMSException e) {
            LOGGER.error("Failed sending message: {}", e.getMessage(), e);
        }
    }
```

메시지를 동기식으로 수신합니다.

```
    /**
     * This method receives messages from a standard Amazon SQS queue using the
Amazon SQS Java
     * Messaging Library. It creates a connection to the queue using JMS (Java
Message Service),
     * waits for messages to arrive, and processes them one at a time. The method
handles all
     * necessary setup and cleanup of messaging resources.
     *
     * @throws JMSException If there is a problem connecting to or receiving
messages from the queue
     */
    public static void doReceiveMessageSync() throws JMSException {
        // Create a connection factory.
        SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
                new ProviderConfiguration(),
                SqsClient.create()
        );

        // Create a connection.
        try (SQSConnection connection = connectionFactory.createConnection() ) {

            // Create the queue if needed.
            SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);

            // Create a session.
```

```
            Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);
            MessageConsumer consumer =
session.createConsumer(session.createQueue(QUEUE_NAME));

            connection.start();

            receiveMessages(consumer);
        }  // The connection closes automatically. This also closes the session.
        LOGGER.info("Connection closed");
    }

    /**
     * This method continuously checks for new messages from a standard Amazon SQS
queue using
     * the Amazon SQS Java Messaging Library. It waits up to 20 seconds for each
message, processes
     * it using JMS (Java Message Service), and confirms receipt. The method stops
checking for
     * messages after 20 seconds of no activity.
     *
     * @param consumer The JMS message consumer that receives messages from the
queue
     */
    private static void receiveMessages(MessageConsumer consumer) {
        try {
            while (true) {
                LOGGER.info("Waiting for messages...");
                // Wait 1 minute for a message
                Message message =
consumer.receive(Duration.ofSeconds(20).toMillis());
                if (message == null) {
                    LOGGER.info("Shutting down after 20 seconds of silence.");
                    break;
                }
                SqsJmsExampleUtils.handleMessage(message);
                message.acknowledge();
                LOGGER.info("Acknowledged message {}", message.getJMSMessageID());
            }
        } catch (JMSException e) {
            LOGGER.error("Error receiving from SQS: {}", e.getMessage(), e);
        }
    }
```

메시지를 비동기식으로 수신합니다.

```
   /**
    * This method sets up automatic message handling for a standard Amazon SQS
queue using the
    * Amazon SQS Java Messaging Library. It creates a listener that processes
messages as soon
    * as they arrive using JMS (Java Message Service), runs for 5 seconds, then
cleans up all
    * messaging resources.
    *
    * @throws JMSException If there is a problem connecting to or receiving
messages from the queue
    */
   public static void doReceiveMessageAsync() throws JMSException {
       // Create a connection factory.
       SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
               new ProviderConfiguration(),
               SqsClient.create()
       );

       // Create a connection.
       try (SQSConnection connection = connectionFactory.createConnection() ) {

           // Create the queue if needed.
           SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);

           // Create a session.
           Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);

           try {
               // Create a consumer for the queue.
               MessageConsumer consumer =
session.createConsumer(session.createQueue(QUEUE_NAME));
               // Provide an implementation of the MessageListener interface, which
has a single 'onMessage' method.
               // We use a lambda expression for the implementation.
               consumer.setMessageListener(message -> {
                   try {
```

```
                    SqsJmsExampleUtils.handleMessage(message);
                    message.acknowledge();
                } catch (JMSException e) {
                    LOGGER.error("Error processing message: {}",
e.getMessage());
                }
            });
            // Start receiving incoming messages.
            connection.start();
            LOGGER.info("Waiting for messages...");
        } catch (JMSException e) {
            throw new RuntimeException(e);
        }
        try {
            Thread.sleep(5000);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }  // The connection closes automatically. This also closes the session.
    LOGGER.info( "Connection closed" );
}
```

CLIENT_ACKNOWLEDGE 모드를 사용하여 메시지를 수신합니다.

```
/**
 * This method demonstrates how message acknowledgment affects message
processing in a standard
 * Amazon SQS queue using the Amazon SQS Java Messaging Library. It sends
messages to the queue,
 * then shows how JMS (Java Message Service) client acknowledgment mode handles
both explicit
 * and implicit message confirmations, including how acknowledging one message
can automatically
 * acknowledge previous messages.
 *
 * @throws JMSException If there is a problem with the messaging operations
 */
public static void doReceiveMessagesSyncClientAcknowledge() throws JMSException
{
    // Create a connection factory.
    SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
            new ProviderConfiguration(),
```

```
                SqsClient.create()
        );

        // Create the connection in a try-with-resources statement so that it's
closed automatically.
        try (SQSConnection connection = connectionFactory.createConnection() ) {

            // Create the queue if needed.
            SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
TIME_OUT_SECONDS);

            // Create a session with client acknowledge mode.
            Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);

            // Create a producer and consumer.
            MessageProducer producer =
session.createProducer(session.createQueue(QUEUE_NAME));
            MessageConsumer consumer =
session.createConsumer(session.createQueue(QUEUE_NAME));

            // Open the connection.
            connection.start();

            // Send two text messages.
            sendMessage(producer, session, "Message 1");
            sendMessage(producer, session, "Message 2");

            // Receive a message and don't acknowledge it.
            receiveMessage(consumer, false);

            // Receive another message and acknowledge it.
            receiveMessage(consumer, true);

            // Wait for the visibility time out, so that unacknowledged messages
reappear in the queue,
            LOGGER.info("Waiting for visibility timeout...");
            try {
                Thread.sleep(TIME_OUT_MILLIS);
            } catch (InterruptedException e) {
                LOGGER.error("Interrupted while waiting for visibility timeout", e);
                Thread.currentThread().interrupt();
                throw new RuntimeException("Processing interrupted", e);
            }
```

```
            /*  We will attempt to receive another message, but none will be
available. This is because in
                CLIENT_ACKNOWLEDGE mode, when we acknowledged the second message,
all previous messages were
                automatically acknowledged as well. Therefore, although we never
directly acknowledged the first
                message, it was implicitly acknowledged when we confirmed the second
one. */
            receiveMessage(consumer, true);
        } // The connection closes automatically. This also closes the session.
        LOGGER.info("Connection closed.");

    }


    /**
     * Sends a text message using the specified JMS MessageProducer and Session.
     *
     * @param producer    The JMS MessageProducer used to send the message
     * @param session     The JMS Session used to create the text message
     * @param messageText The text content to be sent in the message
     * @throws JMSException If there is an error creating or sending the message
     */
    private static void sendMessage(MessageProducer producer, Session session,
String messageText) throws JMSException {
        // Create a text message and send it.
        producer.send(session.createTextMessage(messageText));
    }

    /**
     * Receives and processes a message from a JMS queue using the specified
consumer.
     * The method waits for a message until the configured timeout period is
reached.
     * If a message is received, it is logged and optionally acknowledged based on
the
     * acknowledge parameter.
     *
     * @param consumer    The JMS MessageConsumer used to receive messages from the
queue
     * @param acknowledge Boolean flag indicating whether to acknowledge the
message.
     *                    If true, the message will be acknowledged after processing
```

```
    * @throws JMSException If there is an error receiving, processing, or
acknowledging the message
    */
   private static void receiveMessage(MessageConsumer consumer, boolean
acknowledge) throws JMSException {
       // Receive a message.
       Message message = consumer.receive(TIME_OUT_MILLIS);

       if (message == null) {
           LOGGER.info("Queue is empty!");
       } else {
           // Since this queue has only text messages, cast the message object and
print the text.
           LOGGER.info("Received: {}    Acknowledged: {}", ((TextMessage)
message).getText(), acknowledge);

           // Acknowledge the message if asked.
           if (acknowledge) message.acknowledge();
       }
   }
```

UNORDERED_ACKNOWLEDGE 모드를 사용하여 메시지를 수신합니다.

```
   /**
    * Demonstrates message acknowledgment behavior in UNORDERED_ACKNOWLEDGE mode
with Amazon SQS JMS.
    * In this mode, each message must be explicitly acknowledged regardless of
receive order.
    * Unacknowledged messages return to the queue after the visibility timeout
expires,
    * unlike CLIENT_ACKNOWLEDGE mode where acknowledging one message acknowledges
all previous messages.
    *
    * @throws JMSException         If a JMS-related error occurs during message
operations
    */
   public static void doReceiveMessagesUnorderedAcknowledge() throws JMSException {
       // Create a connection factory.
       SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
               new ProviderConfiguration(),
               SqsClient.create()
       );
```

```
        // Create the connection in a try-with-resources statement so that it's
closed automatically.
        try( SQSConnection connection = connectionFactory.createConnection() ) {

            // Create the queue if needed.
            SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
TIME_OUT_SECONDS);

            // Create a session with unordered acknowledge mode.
            Session session = connection.createSession(false,
SQSSession.UNORDERED_ACKNOWLEDGE);

            // Create the producer and consumer.
            MessageProducer producer =
session.createProducer(session.createQueue(QUEUE_NAME));
            MessageConsumer consumer =
session.createConsumer(session.createQueue(QUEUE_NAME));

            // Open a connection.
            connection.start();

            // Send two text messages.
            sendMessage(producer, session, "Message 1");
            sendMessage(producer, session, "Message 2");

            // Receive a message and don't acknowledge it.
            receiveMessage(consumer, false);

            // Receive another message and acknowledge it.
            receiveMessage(consumer, true);

            // Wait for the visibility time out, so that unacknowledged messages
reappear in the queue.
            LOGGER.info("Waiting for visibility timeout...");
            try {
                Thread.sleep(TIME_OUT_MILLIS);
            } catch (InterruptedException e) {
                LOGGER.error("Interrupted while waiting for visibility timeout", e);
                Thread.currentThread().interrupt();
                throw new RuntimeException("Processing interrupted", e);
            }
```

```
            /*  We will attempt to receive another message, and we'll get the first
message again. This occurs
                because in UNORDERED_ACKNOWLEDGE mode, each message requires its own
separate acknowledgment.
                Since we only acknowledged the second message, the first message
remains in the queue for
                redelivery. */
            receiveMessage(consumer, true);

            LOGGER.info("Connection closed.");
        } // The connection closes automatically. This also closes the session.
    }

    /**
     * Sends a text message to an Amazon SQS queue using JMS.
     *
     * @param producer    The JMS MessageProducer for the queue
     * @param session     The JMS Session for message creation
     * @param messageText The message content
     * @throws JMSException If message creation or sending fails
     */
    private static void sendMessage(MessageProducer producer, Session session,
String messageText) throws JMSException {
        // Create a text message and send it.
        producer.send(session.createTextMessage(messageText));
    }
    /**
     * Synchronously receives a message from an Amazon SQS queue using the JMS API
     * with an acknowledgment parameter.
     *
     * @param consumer    The JMS MessageConsumer for the queue
     * @param acknowledge If true, acknowledges the message after receipt
     * @throws JMSException If message reception or acknowledgment fails
     */
    private static void receiveMessage(MessageConsumer consumer, boolean
acknowledge) throws JMSException {
        // Receive a message.
        Message message = consumer.receive(TIME_OUT_MILLIS);

        if (message == null) {
            LOGGER.info("Queue is empty!");
        } else {
            // Since this queue has only text messages, cast the message object and
print the text.
```

```
            LOGGER.info("Received: {}    Acknowledged: {}", ((TextMessage)
 message).getText(), acknowledge);

            // Acknowledge the message if asked.
            if (acknowledge) message.acknowledge();
        }
    }
```

Spring을 사용하여 종속성을 주입합니다.

```
package com.example.sqs.jms.spring;

import com.amazon.sqs.javamessaging.SQSConnection;
import com.example.sqs.jms.SqsJmsExampleUtils;
import jakarta.jms.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.NoSuchBeanDefinitionException;
import org.springframework.context.support.FileSystemXmlApplicationContext;

import java.io.File;
import java.net.URL;
import java.util.concurrent.TimeUnit;

/**
 * Demonstrates how to send and receive messages using the Amazon SQS Java Messaging
 Library
 * with Spring Framework integration. This example connects to a standard Amazon SQS
 message
 * queue using Spring's dependency injection to configure the connection and
 messaging components.
 * The application uses the JMS (Java Message Service) API to handle message
 operations.
 */
public class SpringExample {
    private static final Integer POLLING_SECONDS = 15;
    private static final String SPRING_XML_CONFIG_FILE =
 "SpringExampleConfiguration.xml.txt";
    private static final Logger LOGGER =
 LoggerFactory.getLogger(SpringExample.class);

    /**
```

```
    * Demonstrates sending and receiving messages through a standard Amazon SQS
message queue
    * using Spring Framework configuration. This method loads connection settings
from an XML file,
    * establishes a messaging session using the Amazon SQS Java Messaging Library,
and processes
    * messages using JMS (Java Message Service) operations. If the queue doesn't
exist, it will
    * be created automatically.
    *
    * @param args Command line arguments (not used)
    */
   public static void main(String[] args) {

       URL resource =
SpringExample.class.getClassLoader().getResource(SPRING_XML_CONFIG_FILE);
       File springFile = new File(resource.getFile());
       if (!springFile.exists() || !springFile.canRead()) {
           LOGGER.error("File " + SPRING_XML_CONFIG_FILE + " doesn't exist or isn't
readable.");
           System.exit(1);
       }

       try (FileSystemXmlApplicationContext context =
                    new FileSystemXmlApplicationContext("file://" +
springFile.getAbsolutePath())) {

           Connection connection;
           try {
               connection = context.getBean(Connection.class);
           } catch (NoSuchBeanDefinitionException e) {
               LOGGER.error("Can't find the JMS connection to use: " +
e.getMessage(), e);
               System.exit(2);
               return;
           }

           String queueName;
           try {
               queueName = context.getBean("queueName", String.class);
           } catch (NoSuchBeanDefinitionException e) {
               LOGGER.error("Can't find the name of the queue to use: " +
e.getMessage(), e);
               System.exit(3);
```

```
                return;
            }
            try {
                if (connection instanceof SQSConnection) {
                    SqsJmsExampleUtils.ensureQueueExists((SQSConnection) connection,
queueName, SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);
                }
                // Create the JMS session.
                Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);

                SqsJmsExampleUtils.sendTextMessage(session, queueName);
                MessageConsumer consumer =
session.createConsumer(session.createQueue(queueName));

                receiveMessages(consumer);
            } catch (JMSException e) {
                LOGGER.error(e.getMessage(), e);
                throw new RuntimeException(e);
            }
        }   // Spring context autocloses. Managed Spring beans that implement
AutoClosable, such as the
        // 'connection' bean, are also closed.
        LOGGER.info("Context closed");
    }

    /**
     * Continuously checks for and processes messages from a standard Amazon SQS
message queue
     * using the Amazon SQS Java Messaging Library underlying the JMS API. This
method waits for incoming messages,
     * processes them when they arrive, and acknowledges their receipt using JMS
(Java Message
     * Service) operations. The method will stop checking for messages after 15
seconds of
     * inactivity.
     *
     * @param consumer The JMS message consumer used to receive messages from the
queue
     */
    private static void receiveMessages(MessageConsumer consumer) {
        try {
            while (true) {
                LOGGER.info("Waiting for messages...");
```

```
                // Wait 15 seconds for a message.
                Message message =
 consumer.receive(TimeUnit.SECONDS.toMillis(POLLING_SECONDS));
                if (message == null) {
                    LOGGER.info("Shutting down after {} seconds of silence.",
 POLLING_SECONDS);
                    break;
                }
                SqsJmsExampleUtils.handleMessage(message);
                message.acknowledge();
                LOGGER.info("Message acknowledged.");
            }
        } catch (JMSException e) {
            LOGGER.error("Error receiving from SQS.", e);
        }
    }
}
```

Spring Bean 정의입니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
        xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="
http://www.springframework.org/schema/beans http://www.springframework.org/schema/
beans/spring-beans-3.0.xsd
        ">
    <!-- Define the AWS Region -->
    <bean id="region" class="software.amazon.awssdk.regions.Region" factory-
method="of">
        <constructor-arg value="us-east-1"/>
    </bean>

    <bean id="credentialsProviderBean"
 class="software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider"
          factory-method="create"/>

    <bean id="clientBuilder" class="software.amazon.awssdk.services.sqs.SqsClient"
 factory-method="builder"/>
```

```
    <bean id="regionSetClientBuilder" factory-bean="clientBuilder" factory-
method="region">
        <constructor-arg ref="region"/>
    </bean>

    <!-- Configure the Builder with Credentials Provider -->
    <bean id="sqsClient" factory-bean="regionSetClientBuilder" factory-
method="credentialsProvider">
        <constructor-arg ref="credentialsProviderBean"/>
    </bean>

    <bean id="providerConfiguration"
 class="com.amazon.sqs.javamessaging.ProviderConfiguration">
        <property name="numberOfMessagesToPrefetch" value="5"/>
    </bean>

    <bean id="connectionFactory"
 class="com.amazon.sqs.javamessaging.SQSConnectionFactory">
        <constructor-arg ref="providerConfiguration"/>
        <constructor-arg ref="clientBuilder"/>
    </bean>

    <bean id="connection"
          factory-bean="connectionFactory"
          factory-method="createConnection"
          init-method="start"
          destroy-method="close"/>

    <bean id="queueName" class="java.lang.String">
        <constructor-arg value="SQSJMSClientExampleQueue"/>
    </bean>
</beans>
```

다른 예제에서 사용하는 일반적인 메서드를 제공하는 유틸리티 클래스입니다.

```
package com.example.sqs.jms;

import com.amazon.sqs.javamessaging.AmazonSQSMessagingClientWrapper;
import com.amazon.sqs.javamessaging.ProviderConfiguration;
import com.amazon.sqs.javamessaging.SQSConnection;
import com.amazon.sqs.javamessaging.SQSConnectionFactory;
```

```
import jakarta.jms.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;

import java.time.Duration;
import java.util.Base64;
import java.util.Map;

/**
 * This utility class provides helper methods for working with Amazon Simple Queue
 Service (Amazon SQS)
 * through the Java Message Service (JMS) interface. It contains common operations
 for managing message
 * queues and handling message delivery.
 */
public class SqsJmsExampleUtils {
    private static final Logger LOGGER =
 LoggerFactory.getLogger(SqsJmsExampleUtils.class);
    public static final Long QUEUE_VISIBILITY_TIMEOUT = 5L;

    /**
     * This method verifies that a message queue exists and creates it if necessary.
 The method checks for
     * an existing queue first to optimize performance.
     *
     * @param connection The active connection to the messaging service
     * @param queueName The name of the queue to verify or create
     * @param visibilityTimeout The duration in seconds that messages will be hidden
 after being received
     * @throws JMSException If there is an error accessing or creating the queue
     */
    public static void ensureQueueExists(SQSConnection connection, String queueName,
 Long visibilityTimeout) throws JMSException {
        AmazonSQSMessagingClientWrapper client =
 connection.getWrappedAmazonSQSClient();

      /* In most cases, you can do this with just a 'createQueue' call, but
 'getQueueUrl'
      (called by 'queueExists') is a faster operation for the common case where the
 queue
```

```
      already exists. Also, many users and roles have permission to call
'getQueueUrl'
      but don't have permission to call 'createQueue'.
      */
       if( !client.queueExists(queueName) ) {
           CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                   .queueName(queueName)
                   .attributes(Map.of(QueueAttributeName.VISIBILITY_TIMEOUT,
String.valueOf(visibilityTimeout)))
                   .build();
           client.createQueue( createQueueRequest );
       }
   }

   /**
    * This method sends a simple text message to a specified message queue. It
handles all necessary
    * setup for the message delivery process.
    *
    * @param session The active messaging session used to create and send the
message
    * @param queueName The name of the queue where the message will be sent
    */
   public static void sendTextMessage(Session session, String queueName) {
       // Rest of implementation...

       try {
           MessageProducer producer =
session.createProducer( session.createQueue( queueName) );
           Message message = session.createTextMessage("Hello world!");
           producer.send(message);
       } catch (JMSException e) {
           LOGGER.error( "Error receiving from SQS", e );
       }
   }

   /**
    * This method processes incoming messages and logs their content based on the
message type.
    * It supports text messages, binary data, and Java objects.
    *
    * @param message The message to be processed and logged
    * @throws JMSException If there is an error reading the message content
    */
```

```
   public static void handleMessage(Message message) throws JMSException {
       // Rest of implementation...
       LOGGER.info( "Got message {}", message.getJMSMessageID() );
       LOGGER.info( "Content: ");
       if(message instanceof TextMessage txtMessage) {
           LOGGER.info( "\t{}", txtMessage.getText() );
       } else if(message instanceof BytesMessage byteMessage){
           // Assume the length fits in an int - SQS only supports sizes up to 256k
so that
           // should be true
           byte[] bytes = new byte[(int)byteMessage.getBodyLength()];
           byteMessage.readBytes(bytes);
           LOGGER.info( "\t{}", Base64.getEncoder().encodeToString( bytes ) );
       } else if( message instanceof ObjectMessage) {
           ObjectMessage objMessage = (ObjectMessage) message;
           LOGGER.info( "\t{}", objMessage.getObject() );
       }
   }

   /**
    * This method sets up automatic message processing for a specified queue. It
creates a listener
    * that will receive and handle incoming messages without blocking the main
program.
    *
    * @param session The active messaging session
    * @param queueName The name of the queue to monitor
    * @param connection The active connection to the messaging service
    */
   public static void receiveMessagesAsync(Session session, String queueName,
Connection connection) {
       // Rest of implementation...
       try {
           // Create a consumer for the queue.
           MessageConsumer consumer =
session.createConsumer(session.createQueue(queueName));
           // Provide an implementation of the MessageListener interface, which has
a single 'onMessage' method.
           // We use a lambda expression for the implementation.
           consumer.setMessageListener(message -> {
               try {
                   SqsJmsExampleUtils.handleMessage(message);
                   message.acknowledge();
               } catch (JMSException e) {
```

```
                    LOGGER.error("Error processing message: {}", e.getMessage());
                }
            });
            // Start receiving incoming messages.
            connection.start();
        } catch (JMSException e) {
            throw new RuntimeException(e);
        }
        try {
            Thread.sleep(2000);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }


    /**
     * This method performs cleanup operations after message processing is complete.
It receives
     * any messages in the specified queue, removes the message queue and closes all
     * active connections to prevent resource leaks.
     *
     * @param queueName The name of the queue to be removed
     * @param visibilityTimeout The duration in seconds that messages are hidden
after being received
     * @throws JMSException If there is an error during the cleanup process
     */
    public static void cleanUpExample(String queueName, Long visibilityTimeout)
throws JMSException {
        LOGGER.info("Performing cleanup.");

        SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
                new ProviderConfiguration(),
                SqsClient.create()
        );

        try (SQSConnection connection = connectionFactory.createConnection() ) {
            ensureQueueExists(connection, queueName, visibilityTimeout);
            Session session = connection.createSession(false,
Session.AUTO_ACKNOWLEDGE);

            receiveMessagesAsync(session, queueName, connection);
```

```
            SqsClient sqsClient =
connection.getWrappedAmazonSQSClient().getAmazonSQSClient();
            try {
                String queueUrl = sqsClient.getQueueUrl(b ->
b.queueName(queueName)).queueUrl();
                sqsClient.deleteQueue(b -> b.queueUrl(queueUrl));
                LOGGER.info("Queue deleted: {}", queueUrl);
            } catch (SdkException e) {
                LOGGER.error("Error during SQS operations: ", e);
            }
        }
        LOGGER.info("Clean up: Connection closed");
    }

    /**
     * This method creates a background task that sends multiple messages to a
specified queue
     * after waiting for a set time period. The task operates independently to
ensure efficient
     * message processing without interrupting other operations.
     *
     * @param queueName The name of the queue where messages will be sent
     * @param secondsToWait The number of seconds to wait before sending messages
     * @param numMessages The number of messages to send
     * @param visibilityTimeout The duration in seconds that messages remain hidden
after being received
     * @return A task that can be executed to send the messages
     */
    public static Runnable sendAMessageAsync(String queueName, Long secondsToWait,
Integer numMessages, Long visibilityTimeout) {
        return () -> {
            try {
                Thread.sleep(Duration.ofSeconds(secondsToWait).toMillis());
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                throw new RuntimeException(e);
            }
            try {
                SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
                        new ProviderConfiguration(),
                        SqsClient.create()
                );
                try (SQSConnection connection =
connectionFactory.createConnection()) {
```

```
                    ensureQueueExists(connection, queueName, visibilityTimeout);
                    Session session = connection.createSession(false,
 Session.CLIENT_ACKNOWLEDGE);
                    for (int i = 1; i <= numMessages; i++) {
                        MessageProducer producer =
 session.createProducer(session.createQueue(queueName));
                        producer.send(session.createTextMessage("Hello World " + i +
 "!"));
                    }
                }
            } catch (JMSException e) {
                LOGGER.error(e.getMessage(), e);
                throw new RuntimeException(e);
            }
        };
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - CreateQueue
  - DeleteQueue

대기열 태그 작업

다음 코드 예제에서는 Amazon SQS에서 태그 지정 작업을 수행하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

다음 예제에서는 대기열에 대한 태그를 생성하고, 태그를 나열하고, 태그를 제거합니다.

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

```
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.ListQueueTagsResponse;
import software.amazon.awssdk.services.sqs.model.QueueDoesNotExistException;
import software.amazon.awssdk.services.sqs.model.SqsException;

import java.util.Map;
import java.util.UUID;

/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials. For more
 * information, see the <a href="https://docs.aws.amazon.com/sdk-for-java/latest/
developer-guide/get-started.html">AWS
 * SDK for Java Developer Guide</a>.
 */
public class TagExamples {
    static final SqsClient sqsClient = SqsClient.create();
    static final String queueName = "TagExamples-queue-" +
 UUID.randomUUID().toString().replace("-", "").substring(0, 20);
    private static final Logger LOGGER = LoggerFactory.getLogger(TagExamples.class);

    public static void main(String[] args) {
        final String queueUrl;
        try {
            queueUrl = sqsClient.createQueue(b ->
 b.queueName(queueName)).queueUrl();
            LOGGER.info("Queue created. The URL is: {}", queueUrl);
        } catch (RuntimeException e) {
            LOGGER.error("Program ending because queue was not created.");
            throw new RuntimeException(e);
        }
        try {
            addTags(queueUrl);
            listTags(queueUrl);
            removeTags(queueUrl);
        } catch (RuntimeException e) {
            LOGGER.error("Program ending because of an error in a method.");
        } finally {
            try {
                sqsClient.deleteQueue(b -> b.queueUrl(queueUrl));
                LOGGER.info("Queue successfully deleted. Program ending.");
                sqsClient.close();
            } catch (RuntimeException e) {
                LOGGER.error("Program ending.");
```

```
            } finally {
                sqsClient.close();
            }
        }
    }

    /** This method demonstrates how to use a Java Map to a tag a aueue.
     * @param queueUrl The URL of the queue to tag.
     */
    public static void addTags(String queueUrl) {
        // Build a map of the tags.
        final Map<String, String> tagsToAdd = Map.of(
                "Team", "Development",
                "Priority", "Beta",
                "Accounting ID", "456def");

        try {
            // Add tags to the queue using a Consumer<TagQueueRequest.Builder>
parameter.
            sqsClient.tagQueue(b -> b
                    .queueUrl(queueUrl)
                    .tags(tagsToAdd)
            );
        } catch (QueueDoesNotExistException e) {
            LOGGER.error("Queue does not exist: {}", e.getMessage(), e);
            throw new RuntimeException(e);
        }
    }

    /** This method demonstrates how to view the tags for a queue.
     * @param queueUrl The URL of the queue whose tags you want to list.
     */
    public static void listTags(String queueUrl) {
        ListQueueTagsResponse response;
        try {
            // Call the listQueueTags method with a
Consumer<ListQueueTagsRequest.Builder> parameter that creates a
ListQueueTagsRequest.
            response = sqsClient.listQueueTags(b -> b
                    .queueUrl(queueUrl));
        } catch (SqsException e) {
            LOGGER.error("Exception thrown: {}", e.getMessage(), e);
            throw new RuntimeException(e);
        }
```

```
        // Log the tags.
        response.tags()
                .forEach((k, v) ->
                        LOGGER.info("Key: {} -> Value: {}", k, v));
    }

    /**
     * This method demonstrates how to remove tags from a queue.
     * @param queueUrl The URL of the queue whose tags you want to remove.
     */
    public static void removeTags(String queueUrl) {
        try {
            // Call the untagQueue method with a Consumer<UntagQueueRequest.Builder>
 parameter.
            sqsClient.untagQueue(b -> b
                    .queueUrl(queueUrl)
                    .tagKeys("Accounting ID") // Remove a single tag.
            );
        } catch (SqsException e) {
            LOGGER.error("Exception thrown: {}", e.getMessage(), e);
            throw new RuntimeException(e);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - ListQueueTags
  - TagQueue
  - UntagQueue

서버리스 예제

Amazon SQS 트리거에서 간접적으로 Lambda 함수 간접 호출

다음 코드 예제는 SQS 대기열에서 메시지를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 함수는 이벤트 파라미터에서 메시지를 검색하고 각 메시지의 내용을 로깅합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Java를 사용하여 Lambda로 SQS 이벤트 사용

```
// 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;
        }

    }
}
```

Amazon SQS 트리거로 Lambda 함수에 대한 배치 항목 실패 보고

다음 코드 예제는 SQS 대기열에서 이벤트를 수신하는 Lambda 함수에 대한 부분 배치 응답을 구현하는 방법을 보여줍니다. 이 함수는 응답으로 배치 항목 실패를 보고하고 나중에 해당 메시지를 다시 시도하도록 Lambda에 신호를 보냅니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Java를 사용하여 Lambda로 SQS 배치 항목 실패 보고

```
// 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>();

        for (SQSEvent.SQSMessage message : sqsEvent.getRecords()) {
            try {
                //process your message
            } catch (Exception e) {
                //Add failed message identifier to the batchItemFailures list
                batchItemFailures.add(new
 SQSBatchResponse.BatchItemFailure(message.getMessageId()));
            }
        }
```

```
            return new SQSBatchResponse(batchItemFailures);
        }
}
```

Java 2.x용 SDK를 사용하는 Step Functions의 예제

다음 코드 예제에서는 Step Functions와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시작하기

Hello Step Functions

다음 코드 예제에서는 Step Functions 사용을 시작하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Hello의 Java 버전.

```
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);
        }
    }
```

```
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListStateMachines를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 활동을 생성합니다.
- 이전에 생성한 활동을 한 단계로 포함하는 Amazon States Language 정의에서 상태 시스템을 생성합니다.
- 상태 시스템을 실행하고 사용자 입력으로 활동에 응답합니다.
- 실행 완료 후 최종 상태 및 출력을 가져온 다음 리소스를 정리합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * You can obtain the JSON file to create a state machine in the following
 * GitHub location.
 * <p>
 * https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/resources/sample_files
 * <p>
 * To run this code example, place the chat_sfn_state_machine.json file into
 * your project's resources folder.
 * <p>
 * Also, set up your development environment, including your credentials.
 * <p>
 * For information, see this documentation topic:
 * <p>
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 * <p>
 * This Java code example performs the following tasks:
 * <p>
 * 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.
                jsonFile - The location of the chat_sfn_state_machine.json file. You
 can located it in resources/sample_files.
            """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleName = args[0];
        String activityName = args[1];
        String stateMachineName = args[2];
        String jsonFile = args[3];
        String polJSON = """
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
```

```
                    "Sid": "",
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "states.amazonaws.com"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        }
        """;


        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);

        // Read the file using FileInputStream
        FileInputStream inputStream = new FileInputStream(jsonFile);
        ObjectMapper mapper = new ObjectMapper();
        JsonNode jsonNode = mapper.readValue(inputStream, 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 "";
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - CreateActivity
  - CreateStateMachine
  - DeleteActivity
  - DeleteStateMachine
  - DescribeExecution
  - DescribeStateMachine
  - GetActivityTask
  - ListActivities
  - ListStateMachines
  - SendTaskSuccess
  - StartExecution
  - StopExecution

작업

`CreateActivity`

다음 코드 예시는 `CreateActivity`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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 "";
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateActivity를 참조하세요.

`CreateStateMachine`

다음 코드 예시는 `CreateStateMachine`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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 "";
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateStateMachine을 참조하세요.

`DeleteActivity`

다음 코드 예시는 `DeleteActivity`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteActivity를 참조하세요.

`DeleteStateMachine`

다음 코드 예시는 `DeleteStateMachine`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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.");
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteStateMachine을 참조하세요.

`DescribeExecution`

다음 코드 예시는 `DescribeExecution`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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);
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeExecution을 참조하세요.

`DescribeStateMachine`

다음 코드 예시는 `DescribeStateMachine`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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());
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeStateMachine을 참조하세요.

`GetActivityTask`

다음 코드 예시는 `GetActivityTask`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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;
```

```
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 GetActivityTask를 참조하세요.

`ListActivities`

다음 코드 예시는 `ListActivities`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API Reference의 ListActivities를 참조하세요.

`ListExecutions`

다음 코드 예시는 `ListExecutions`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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);
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API Reference의 ListExecutions를 참조하세요.

`ListStateMachines`

다음 코드 예시는 `ListStateMachines`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 ListStateMachines를 참조하세요.

SendTaskSuccess

다음 코드 예시는 SendTaskSuccess의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 SendTaskSuccess를 참조하세요.

`StartExecution`

다음 코드 예시는 `StartExecution`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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 "";
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 StartExecution을 참조하세요.

시나리오

Step Functions를 사용하여 Lambda 함수 간접 호출

다음 코드 예제에서는 AWS Lambda 함수를 순차적으로 호출하는 AWS Step Functions 상태 시스템을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

AWS Step Functions 및를 사용하여 AWS 서버리스 워크플로를 생성하는 방법을 보여줍니다 AWS SDK for Java 2.x. 각 워크플로 단계는 AWS Lambda 함수를 사용하여 구현됩니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- DynamoDB

- Lambda
- Amazon SES
- 단계 함수

AWS STS SDK for Java 2.x를 사용한 예제

다음 코드 예제에서는와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS STS.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`AssumeRole`

다음 코드 예시는 `AssumeRole`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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, arn: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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 AssumeRole을 참조하세요.

지원 SDK for Java 2.x를 사용한 예제

다음 코드 예제에서는를와 AWS SDK for Java 2.x 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 지원.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시작하기
- 기본 사항
- 작업

시작하기

안녕하세요 지원

다음 코드 예제에서는 지원를 사용하여 시작하는 방법을 보여 줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeServices를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 사용 가능한 서비스 및 사례의 심각도 수준을 가져와서 표시합니다.
- 선택한 서비스, 범주 및 심각도 수준을 사용하여 지원 사례를 만듭니다.
- 현재 일자의 미해결 사례 목록을 가져와서 표시합니다.
- 새로운 사례에 첨부 파일 세트와 통신을 추가합니다.
- 해당 사례에 대한 새로운 첨부 파일과 통신을 설명하세요.
- 사건을 해결하세요.
- 현재 일자의 해결된 사례 목록을 가져와서 표시합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

다양한 지원 작업을 실행합니다.

```
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 = "C:\\AWS\\test.txt" ; //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;
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - AddAttachmentsToSet
  - AddCommunicationToCase
  - CreateCase
  - DescribeAttachment
  - DescribeCases
  - DescribeCommunications
  - DescribeServices

- DescribeSeverityLevels
- ResolveCase

작업

`AddAttachmentsToSet`

다음 코드 예시는 `AddAttachmentsToSet`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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 "";
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 AddAttachmentsToSet를 참조하세요.

`AddCommunicationToCase`

다음 코드 예시는 `AddCommunicationToCase`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
```

```
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 AddCommunicationToCase를 참조하세요.

`CreateCase`

다음 코드 예시는 `CreateCase`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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 "";
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateCase를 참조하세요.

`DescribeAttachment`

다음 코드 예시는 `DescribeAttachment`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeAttachment를 참조하세요.

`DescribeCases`

다음 코드 예시는 `DescribeCases`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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);
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeCases를 참조하세요.

`DescribeCommunications`

다음 코드 예시는 `DescribeCommunications`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   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 "";
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeCommunications를 참조하세요.

`DescribeServices`

다음 코드 예시는 `DescribeServices`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   // 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;
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeServices를 참조하세요.

`DescribeSeverityLevels`

다음 코드 예시는 `DescribeSeverityLevels`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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 "";
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeSeverityLevels를 참조하세요.

`ResolveCase`

다음 코드 예시는 `ResolveCase`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    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);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ResolveCase를 참조하세요.

Java 2.x용 SDK를 사용하는 Secrets Manager 예제

다음 코드 예제에서는 Systems Manager와 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시작하기
- 기본 사항
- 작업

시작하기

Hello Systems Manager

다음 코드 예제에서는 Systems Manager를 사용하여 시작하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListDocuments를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 유지 관리 기간을 생성합니다.
- 유지 보수 기간 일정을 수정합니다.
- 문서를 만듭니다.

- 지정된 EC2 인스턴스로 명령을 보냅니다.
- OpsItem을 생성합니다.
- OpsItem을 업데이트하고 해결합니다.
- 유지 보수 기간, OpsItem 및 문서를 삭제합니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.services.ssm.model.DocumentAlreadyExistsException;
import software.amazon.awssdk.services.ssm.model.SsmException;

import java.util.Map;
import java.util.Scanner;
public class SSMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final String ROLES_STACK = "SsmStack3`1";

    public static void main(String[] args) {
        String usage = """
            Usage:
              <title> <source> <category> <severity>

            Where:
                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).
        """;

        Scanner scanner = new Scanner(System.in);
        SSMActions actions = new SSMActions();
        String documentName;
```

```
        String windowName;

        System.out.println("Use AWS CloudFormation to create the EC2 instance that
is required for this scenario.");
        CloudFormationHelper.deployCloudFormationStack(ROLES_STACK);
        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputsAsync(ROLES_STACK).join();
        String instanceId = stackOutputs.get("InstanceId");
        System.out.println("The Instance ID: " + instanceId +" was created.");
        String title = "Disk Space Alert" ;
        String source = "EC2" ;
        String category = "Availability" ;
        String severity = "2" ;

        System.out.println(DASHES);
        System.out.println("""
                Welcome to the AWS Systems Manager SDK Basics scenario.
                This Java program demonstrates how to interact with AWS Systems
Manager using the AWS SDK for Java (v2).
                AWS Systems Manager is the operations hub for your AWS applications
and resources and a secure end-to-end management solution.
                The program's primary functionalities 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 AWS SSM resources.
                Upon completion of the program, all AWS resources are cleaned up.
                Let's get started...

                """);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("1. Create an SSM 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 = null;
        try {
            winId = actions.createMaintenanceWindow(windowName);
            waitForInputToContinue(scanner);
            System.out.println("The maintenance window ID is: " + winId);
        } catch (DocumentAlreadyExistsException e) {
```

```
            System.err.println("The SSM maintenance window already exists.
Retrieving existing window ID...");
            String existingWinId = actions.createMaintenanceWindow(windowName);
            System.out.println("Existing window ID: " + existingWinId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("2. Modify the maintenance window by changing the
schedule");
        waitForInputToContinue(scanner);
        try {
            actions.updateSSMMaintenanceWindow(winId, windowName);
            waitForInputToContinue(scanner);
            System.out.println("The SSM maintenance window was successfully
updated");
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("3. Create an SSM document that defines the actions that
Systems Manager performs on your managed nodes.");
        System.out.println("Please enter the document name (default is
ssmdocument):");
        String doc = scanner.nextLine();
        documentName = doc.isEmpty() ? "ssmdocument" : doc;
        try {
            actions.createSSMDoc(documentName);
            waitForInputToContinue(scanner);
            System.out.println("The SSM document was successfully created");
        } catch (DocumentAlreadyExistsException e) {
            System.err.println("The SSM document already exists. Moving on");
```

```
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("4. Now we are going to run a command on an EC2
instance");
        waitForInputToContinue(scanner);
        String commandId="";
        try {
            commandId = actions.sendSSMCommand(documentName, instanceId);
            waitForInputToContinue(scanner);
            System.out.println("The command was successfully sent. Command ID: " +
commandId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
        } catch (InterruptedException e) {
            System.err.println("Thread was interrupted: " + e.getMessage());
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("5. Lets get the time when the specific command was sent
to the specific managed node");
        waitForInputToContinue(scanner);
        try {
            actions.displayCommands(commandId);
            System.out.println("The command invocations were successfully
displayed.");
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("""
             6. Now we will create an SSM OpsItem.
             A SSM OpsItem is a feature provided by Amazon's Systems Manager (SSM)
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.
            """);

        waitForInputToContinue(scanner);
        String opsItemId;
        try {
            opsItemId = actions.createSSMOpsItem(title, source, category, severity);
            System.out.println(opsItemId + " was created");
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Now we will update the SSM OpsItem "+opsItemId);
        waitForInputToContinue(scanner);
        String description = "An update to "+opsItemId ;
        try {
            actions.updateOpsItem(opsItemId, title, description);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
```

```
        }

        System.out.println(DASHES);
        System.out.println("8. Now we will get the status of the SSM OpsItem
"+opsItemId);
        waitForInputToContinue(scanner);
        try {
            actions.describeOpsItems(opsItemId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }

        System.out.println(DASHES);
        System.out.println("9. Now we will resolve the SSM OpsItem "+opsItemId);
        waitForInputToContinue(scanner);
        try {
            actions.resolveOpsItem(opsItemId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }

        System.out.println(DASHES);
        System.out.println("10. Would you like to delete the AWS Systems Manager
resources? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            System.out.println("You selected to delete the resources.");
            waitForInputToContinue(scanner);
            try {
                actions.deleteMaintenanceWindow(winId);
                actions.deleteDoc(documentName);
            } catch (SsmException e) {
                System.err.println("SSM error: " + e.getMessage());
                return;
            } catch (RuntimeException e) {
                System.err.println("Unexpected error: " + e.getMessage());
```

```
                return;
            }
        } else {
            System.out.println("The AWS Systems Manager resources will not be
 deleted");
        }
        System.out.println(DASHES);
        CloudFormationHelper.destroyCloudFormationStack(ROLES_STACK);
        System.out.println("This concludes the AWS Systems Manager SDK Basics
 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.");
            }
        }
    }
}
```

Systems Manager SDK 메서드에 대한 래퍼 클래스입니다.

```
public class SSMActions {

    private static SsmAsyncClient ssmAsyncClient;

    private static SsmAsyncClient getAsyncClient() {
        if (ssmAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                    .maxConcurrency(100)
                    .connectionTimeout(Duration.ofSeconds(60))
```

```
                    .readTimeout(Duration.ofSeconds(60))
                    .writeTimeout(Duration.ofSeconds(60))
                    .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
                    .retryPolicy(RetryPolicy.builder()
                            .numRetries(3)
                            .build())
                    .build();

            ssmAsyncClient = SsmAsyncClient.builder()
                    .region(Region.US_WEST_2)
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
        return ssmAsyncClient;
    }

    /**
     * Deletes an AWS SSM document asynchronously.
     *
     * @param documentName The name of the document to delete.
     * <p>
     * This method initiates an asynchronous request to delete an SSM document.
     * If an exception occurs, it handles the error appropriately.
     */
    public void deleteDoc(String documentName) {
        DeleteDocumentRequest documentRequest = DeleteDocumentRequest.builder()
                .name(documentName)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().deleteDocument(documentRequest)
                    .thenAccept(response -> {
                        System.out.println("The SSM document was successfully
deleted.");
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
```

```
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }

    /**
     * Deletes an AWS SSM Maintenance Window asynchronously.
     *
     * @param winId The ID of the Maintenance Window to delete.
     * <p>
     * This method initiates an asynchronous request to delete an SSM Maintenance
Window.
     * If an exception occurs, it handles the error appropriately.
     */
    public void deleteMaintenanceWindow(String winId) {
        DeleteMaintenanceWindowRequest windowRequest =
DeleteMaintenanceWindowRequest.builder()
                .windowId(winId)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().deleteMaintenanceWindow(windowRequest)
                    .thenAccept(response -> {
                        System.out.println("The maintenance window was successfully
deleted.");
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
```

```
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }

    /**
     * Resolves an AWS SSM OpsItem asynchronously.
     *
     * @param opsID The ID of the OpsItem to resolve.
     * <p>
     * This method initiates an asynchronous request to resolve an SSM OpsItem.
     * If an exception occurs, it handles the error appropriately.
     */
    public void resolveOpsItem(String opsID) {
        UpdateOpsItemRequest opsItemRequest = UpdateOpsItemRequest.builder()
                .opsItemId(opsID)
                .status(OpsItemStatus.RESOLVED)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().updateOpsItem(opsItemRequest)
                    .thenAccept(response -> {
                        System.out.println("OpsItem resolved successfully.");
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
```

```
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }

    /**
     * Describes AWS SSM OpsItems asynchronously.
     *
     * @param key The key to filter OpsItems by (e.g., OPS_ITEM_ID).
     *
     * This method initiates an asynchronous request to describe SSM OpsItems.
     * If the request is successful, it prints the title and status of each OpsItem.
     * If an exception occurs, it handles the error appropriately.
     */
    public void describeOpsItems(String key) {
        OpsItemFilter filter = OpsItemFilter.builder()
                .key(OpsItemFilterKey.OPS_ITEM_ID)
                .values(key)
                .operator(OpsItemFilterOperator.EQUAL)
                .build();

        DescribeOpsItemsRequest itemsRequest = DescribeOpsItemsRequest.builder()
                .maxResults(10)
                .opsItemFilters(filter)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().describeOpsItems(itemsRequest)
                    .thenAccept(itemsResponse -> {
```

```
                        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());
                        }
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }

    /**
     * Updates the AWS SSM OpsItem asynchronously.
     *
     * @param opsItemId The ID of the OpsItem to update.
     * @param title The new title of the OpsItem.
     * @param description The new description of the OpsItem.
     * <p>
     * This method initiates an asynchronous request to update an SSM OpsItem.
     * If the request is successful, it completes without returning a value.
     * If an exception occurs, it handles the error appropriately.
     */
    public void updateOpsItem(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());

        CompletableFuture<Void> future = getOpsItem(opsItemId).thenCompose(opsItem -
> {
            UpdateOpsItemRequest request = UpdateOpsItemRequest.builder()
                    .opsItemId(opsItemId)
                    .title(title)
                    .operationalData(operationalData)
                    .status(opsItem.statusAsString())
                    .description(description)
                    .build();

            return getAsyncClient().updateOpsItem(request).thenAccept(response -> {
                System.out.println(opsItemId + " updated successfully.");
            }).exceptionally(ex -> {
                throw new CompletionException(ex);
            });
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
 ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
 cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
 cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
 ex.getCause() : ex;
        }
    }


    private static CompletableFuture<OpsItem> getOpsItem(String opsItemId) {
        GetOpsItemRequest request =
 GetOpsItemRequest.builder().opsItemId(opsItemId).build();
```

```
        return
getAsyncClient().getOpsItem(request).thenApply(GetOpsItemResponse::opsItem);
    }

    /**
     * Creates an SSM OpsItem asynchronously.
     *
     * @param title The title of the OpsItem.
     * @param source The source of the OpsItem.
     * @param category The category of the OpsItem.
     * @param severity The severity of the OpsItem.
     * @return The ID of the created OpsItem.
     * <p>
     * This method initiates an asynchronous request to create an SSM OpsItem.
     * If the request is successful, it returns the OpsItem ID.
     * If an exception occurs, it handles the error appropriately.
     */
    public String createSSMOpsItem(String title, String source, String category,
String severity) {
        CreateOpsItemRequest opsItemRequest = CreateOpsItemRequest.builder()
                .description("Created by the SSM Java API")
                .title(title)
                .source(source)
                .category(category)
                .severity(severity)
                .build();

        CompletableFuture<CreateOpsItemResponse> future =
getAsyncClient().createOpsItem(opsItemRequest);

        try {
            CreateOpsItemResponse response = future.join();
            return response.opsItemId();
        } catch (CompletionException e) {
            Throwable cause = e.getCause();
            if (cause instanceof SsmException) {
                throw (SsmException) cause;
            } else {
                throw new RuntimeException(cause);
            }
        }
    }

    /**
```

```
    * Displays the date and time when the specific command was invoked.
    *
    * @param commandId The ID of the command to describe.
    * <p>
    * This method initiates an asynchronous request to list command invocations and
prints the date and time of each command invocation.
    * If an exception occurs, it handles the error appropriately.
    */
   public void displayCommands(String commandId) {
       ListCommandInvocationsRequest commandInvocationsRequest =
ListCommandInvocationsRequest.builder()
               .commandId(commandId)
               .build();

       CompletableFuture<ListCommandInvocationsResponse> future =
getAsyncClient().listCommandInvocations(commandInvocationsRequest);
       future.thenAccept(response -> {
           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()));
           }
       }).exceptionally(ex -> {
           Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
           if (cause instanceof SsmException) {
               throw (SsmException) cause;
           } else {
               throw new RuntimeException(cause);
           }
       }).join();
   }

   /**
    * Sends a SSM command to a managed node asynchronously.
    *
    * @param documentName The name of the document to use.
    * @param instanceId The ID of the instance to send the command to.
    * @return The command ID.
    * <p>
    * This method initiates asynchronous requests to send a SSM command to a
managed node.
```

```
    * It waits until the document is active, sends the command, and checks the
command execution status.
    */
   public String sendSSMCommand(String documentName, String instanceId) throws
InterruptedException, SsmException {
       // Before we use Document to send a command - make sure it is active.
       CompletableFuture<Void> documentActiveFuture = CompletableFuture.runAsync(()
-> {
           boolean isDocumentActive = false;
           DescribeDocumentRequest request = DescribeDocumentRequest.builder()
                   .name(documentName)
                   .build();

           while (!isDocumentActive) {
               CompletableFuture<DescribeDocumentResponse> response =
getAsyncClient().describeDocument(request);
               String documentStatus = response.join().document().statusAsString();
               if (documentStatus.equals("Active")) {
                   System.out.println("The SSM document is active and ready to
use.");
                   isDocumentActive = true;
               } else {
                   System.out.println("The SSM document is not active. Status: " +
documentStatus);
                   try {
                       Thread.sleep(5000);
                   } catch (InterruptedException e) {
                       throw new RuntimeException(e);
                   }
               }
           }
       });

       documentActiveFuture.join();

       // Create the SendCommandRequest.
       SendCommandRequest commandRequest = SendCommandRequest.builder()
               .documentName(documentName)
               .instanceIds(instanceId)
               .build();

       // Send the command.
       CompletableFuture<SendCommandResponse> commandFuture =
getAsyncClient().sendCommand(commandRequest);
```

```
        final String[] commandId = {null};

        commandFuture.whenComplete((commandResponse, ex) -> {
            if (commandResponse != null) {
                commandId[0] = commandResponse.command().commandId();
                System.out.println("Command ID: " + commandId[0]);

                // Wait for the command execution to complete.
                GetCommandInvocationRequest invocationRequest =
GetCommandInvocationRequest.builder()
                        .commandId(commandId[0])
                        .instanceId(instanceId)
                        .build();

                try {
                    System.out.println("Wait 5 secs");
                    TimeUnit.SECONDS.sleep(5);

                    // Retrieve the command execution details.
                    CompletableFuture<GetCommandInvocationResponse> invocationFuture
= getAsyncClient().getCommandInvocation(invocationRequest);
                    invocationFuture.whenComplete((commandInvocationResponse,
invocationEx) -> {
                        if (commandInvocationResponse != null) {
                            // 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);
                            }
                        } else {
                            Throwable invocationCause = (invocationEx instanceof
CompletionException) ? invocationEx.getCause() : invocationEx;
                            throw new CompletionException(invocationCause);
                        }
                    }).join();
                } catch (InterruptedException e) {
                    throw new RuntimeException(e);
                }
            } else {
```

```
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof SsmException) {
                    throw (SsmException) cause;
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();

        return commandId[0];
    }

    /**
     * Creates an AWS SSM document asynchronously.
     *
     * @param docName The name of the document to create.
     * <p>
     * This method initiates an asynchronous request to create an SSM document.
     * If the request is successful, it prints the document status.
     * If an exception occurs, it handles the error appropriately.
     */
    public void createSSMDoc(String docName) throws SsmException {
        String jsonData = """
        {
        "schemaVersion": "2.2",
        "description": "Run a simple shell command",
        "mainSteps": [
            {
                "action": "aws:runShellScript",
                "name": "runEchoCommand",
                "inputs": {
                  "runCommand": [
                    "echo 'Hello, world!'"
                  ]
                }
              }
            ]
        }
        """;

        CreateDocumentRequest request = CreateDocumentRequest.builder()
                .content(jsonData)
                .name(docName)
```

```
                .documentType(DocumentType.COMMAND)
                .build();

        CompletableFuture<CreateDocumentResponse> future =
getAsyncClient().createDocument(request);
        future.thenAccept(response -> {
            System.out.println("The status of the SSM document is " +
response.documentDescription().status());
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof DocumentAlreadyExistsException) {
                throw new CompletionException(cause);
            } else if (cause instanceof SsmException) {
                throw new CompletionException(cause);
            } else {
                throw new RuntimeException(cause);
            }
        }).join();
    }

    /**
     * Updates an SSM maintenance window asynchronously.
     *
     * @param id The ID of the maintenance window to update.
     * @param name The new name for the maintenance window.
     * <p>
     * This method initiates an asynchronous request to update an SSM maintenance
window.
     * If the request is successful, it prints a success message.
     * If an exception occurs, it handles the error appropriately.
     */
    public void updateSSMMaintenanceWindow(String id, String name) throws
SsmException {
        UpdateMaintenanceWindowRequest updateRequest =
UpdateMaintenanceWindowRequest.builder()
                .windowId(id)
                .allowUnassociatedTargets(true)
                .duration(24)
                .enabled(true)
                .name(name)
                .schedule("cron(0 0 ? * MON *)")
                .build();
```

```
        CompletableFuture<UpdateMaintenanceWindowResponse> future =
getAsyncClient().updateMaintenanceWindow(updateRequest);
        future.whenComplete((response, ex) -> {
            if (response != null) {
                System.out.println("The SSM maintenance window was successfully
updated");
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof SsmException) {
                    throw new CompletionException(cause);
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();
    }

    /**
     * Creates an SSM maintenance window asynchronously.
     *
     * @param winName The name of the maintenance window.
     * @return The ID of the created or existing maintenance window.
     * <p>
     * This method initiates an asynchronous request to create an SSM maintenance
window.
     * If the request is successful, it prints the maintenance window ID.
     * If an exception occurs, it handles the error appropriately.
     */
    public String createMaintenanceWindow(String winName) throws SsmException,
DocumentAlreadyExistsException {
        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();

        CompletableFuture<CreateMaintenanceWindowResponse> future =
getAsyncClient().createMaintenanceWindow(request);
        final String[] windowId = {null};
```

```
        future.whenComplete((response, ex) -> {
            if (response != null) {
                String maintenanceWindowId = response.windowId();
                System.out.println("The maintenance window id is " +
maintenanceWindowId);
                windowId[0] = maintenanceWindowId;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof DocumentAlreadyExistsException) {
                    throw new CompletionException(cause);
                } else if (cause instanceof SsmException) {
                    throw new CompletionException(cause);
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();

        if (windowId[0] == null) {
            MaintenanceWindowFilter filter = MaintenanceWindowFilter.builder()
                    .key("name")
                    .values(winName)
                    .build();

            DescribeMaintenanceWindowsRequest winRequest =
DescribeMaintenanceWindowsRequest.builder()
                    .filters(filter)
                    .build();

            CompletableFuture<DescribeMaintenanceWindowsResponse> describeFuture =
getAsyncClient().describeMaintenanceWindows(winRequest);
            describeFuture.whenComplete((describeResponse, describeEx) -> {
                if (describeResponse != null) {
                    List<MaintenanceWindowIdentity> windows =
describeResponse.windowIdentities();
                    if (!windows.isEmpty()) {
                        windowId[0] = windows.get(0).windowId();
                        System.out.println("Window ID: " + windowId[0]);
                    } else {
                        System.out.println("Window not found.");
                        windowId[0] = "";
                    }
                } else {
```

```
                    Throwable describeCause = (describeEx instanceof
 CompletionException) ? describeEx.getCause() : describeEx;
                    throw new RuntimeException("Error describing maintenance
 windows: " + describeCause.getMessage(), describeCause);
                }
            }).join();
        }

        return windowId[0];
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - CreateDocument
  - CreateMaintenanceWindow
  - CreateOpsItem
  - DeleteMaintenanceWindow
  - ListCommandInvocations
  - SendCommand
  - UpdateOpsItem

작업

`CreateDocument`

다음 코드 예시는 `CreateDocument`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Creates an AWS SSM document asynchronously.
```

```
     *
     * @param docName The name of the document to create.
     * <p>
     * This method initiates an asynchronous request to create an SSM document.
     * If the request is successful, it prints the document status.
     * If an exception occurs, it handles the error appropriately.
     */
    public void createSSMDoc(String docName) throws SsmException {
        String jsonData = """
        {
        "schemaVersion": "2.2",
        "description": "Run a simple shell command",
        "mainSteps": [
            {
                "action": "aws:runShellScript",
                "name": "runEchoCommand",
                "inputs": {
                  "runCommand": [
                    "echo 'Hello, world!'"
                  ]
                }
              }
            ]
        }
        """;

        CreateDocumentRequest request = CreateDocumentRequest.builder()
                .content(jsonData)
                .name(docName)
                .documentType(DocumentType.COMMAND)
                .build();

        CompletableFuture<CreateDocumentResponse> future =
getAsyncClient().createDocument(request);
        future.thenAccept(response -> {
            System.out.println("The status of the SSM document is " +
response.documentDescription().status());
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof DocumentAlreadyExistsException) {
                throw new CompletionException(cause);
            } else if (cause instanceof SsmException) {
                throw new CompletionException(cause);
```

```
            } else {
                throw new RuntimeException(cause);
            }
        }).join();
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateDocument를 참조하세요.

`CreateMaintenanceWindow`

다음 코드 예시는 `CreateMaintenanceWindow`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Creates an SSM maintenance window asynchronously.
     *
     * @param winName The name of the maintenance window.
     * @return The ID of the created or existing maintenance window.
     * <p>
     * This method initiates an asynchronous request to create an SSM maintenance
window.
     * If the request is successful, it prints the maintenance window ID.
     * If an exception occurs, it handles the error appropriately.
     */
    public String createMaintenanceWindow(String winName) throws SsmException,
DocumentAlreadyExistsException {
        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();

        CompletableFuture<CreateMaintenanceWindowResponse> future =
getAsyncClient().createMaintenanceWindow(request);
        final String[] windowId = {null};
        future.whenComplete((response, ex) -> {
            if (response != null) {
                String maintenanceWindowId = response.windowId();
                System.out.println("The maintenance window id is " +
maintenanceWindowId);
                windowId[0] = maintenanceWindowId;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof DocumentAlreadyExistsException) {
                    throw new CompletionException(cause);
                } else if (cause instanceof SsmException) {
                    throw new CompletionException(cause);
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();

        if (windowId[0] == null) {
            MaintenanceWindowFilter filter = MaintenanceWindowFilter.builder()
                    .key("name")
                    .values(winName)
                    .build();

            DescribeMaintenanceWindowsRequest winRequest =
DescribeMaintenanceWindowsRequest.builder()
                    .filters(filter)
                    .build();

            CompletableFuture<DescribeMaintenanceWindowsResponse> describeFuture =
getAsyncClient().describeMaintenanceWindows(winRequest);
            describeFuture.whenComplete((describeResponse, describeEx) -> {
                if (describeResponse != null) {
                    List<MaintenanceWindowIdentity> windows =
describeResponse.windowIdentities();
                    if (!windows.isEmpty()) {
                        windowId[0] = windows.get(0).windowId();
                        System.out.println("Window ID: " + windowId[0]);
```

```
                    } else {
                        System.out.println("Window not found.");
                        windowId[0] = "";
                    }
                } else {
                    Throwable describeCause = (describeEx instanceof
CompletionException) ? describeEx.getCause() : describeEx;
                    throw new RuntimeException("Error describing maintenance
windows: " + describeCause.getMessage(), describeCause);
                }
            }).join();
        }

        return windowId[0];
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateMaintenanceWindow를 참조하세요.

`CreateOpsItem`

다음 코드 예시는 `CreateOpsItem`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Creates an SSM OpsItem asynchronously.
     *
     * @param title The title of the OpsItem.
     * @param source The source of the OpsItem.
     * @param category The category of the OpsItem.
     * @param severity The severity of the OpsItem.
     * @return The ID of the created OpsItem.
     * <p>
     * This method initiates an asynchronous request to create an SSM OpsItem.
     * If the request is successful, it returns the OpsItem ID.
```

```
     * If an exception occurs, it handles the error appropriately.
     */
    public String createSSMOpsItem(String title, String source, String category,
String severity) {
        CreateOpsItemRequest opsItemRequest = CreateOpsItemRequest.builder()
                .description("Created by the SSM Java API")
                .title(title)
                .source(source)
                .category(category)
                .severity(severity)
                .build();

        CompletableFuture<CreateOpsItemResponse> future =
getAsyncClient().createOpsItem(opsItemRequest);

        try {
            CreateOpsItemResponse response = future.join();
            return response.opsItemId();
        } catch (CompletionException e) {
            Throwable cause = e.getCause();
            if (cause instanceof SsmException) {
                throw (SsmException) cause;
            } else {
                throw new RuntimeException(cause);
            }
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 CreateOpsItem을 참조하세요.

`DeleteDocument`

다음 코드 예시는 `DeleteDocument`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Deletes an AWS SSM document asynchronously.
    *
    * @param documentName The name of the document to delete.
    * <p>
    * This method initiates an asynchronous request to delete an SSM document.
    * If an exception occurs, it handles the error appropriately.
    */
   public void deleteDoc(String documentName) {
       DeleteDocumentRequest documentRequest = DeleteDocumentRequest.builder()
               .name(documentName)
               .build();

       CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
           getAsyncClient().deleteDocument(documentRequest)
                   .thenAccept(response -> {
                       System.out.println("The SSM document was successfully
deleted.");
                   })
                   .exceptionally(ex -> {
                       throw new CompletionException(ex);
                   }).join();
       }).exceptionally(ex -> {
           Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
           if (cause instanceof SsmException) {
               throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
           } else {
               throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
           }
       });

       try {
           future.join();
       } catch (CompletionException ex) {
           throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
       }
   }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteDocument를 참조하세요.

`DeleteMaintenanceWindow`

다음 코드 예시는 `DeleteMaintenanceWindow`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Deletes an AWS SSM Maintenance Window asynchronously.
    *
    * @param winId The ID of the Maintenance Window to delete.
    * <p>
    * This method initiates an asynchronous request to delete an SSM Maintenance
Window.
    * If an exception occurs, it handles the error appropriately.
    */
   public void deleteMaintenanceWindow(String winId) {
       DeleteMaintenanceWindowRequest windowRequest =
DeleteMaintenanceWindowRequest.builder()
               .windowId(winId)
               .build();

       CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
           getAsyncClient().deleteMaintenanceWindow(windowRequest)
                   .thenAccept(response -> {
                       System.out.println("The maintenance window was successfully
deleted.");
                   })
                   .exceptionally(ex -> {
                       throw new CompletionException(ex);
                   }).join();
       }).exceptionally(ex -> {
           Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
           if (cause instanceof SsmException) {
```

```
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 DeleteMaintenanceWindow를 참조하세요.

`DescribeOpsItems`

다음 코드 예시는 `DescribeOpsItems`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Describes AWS SSM OpsItems asynchronously.
     *
     * @param key The key to filter OpsItems by (e.g., OPS_ITEM_ID).
     *
     * This method initiates an asynchronous request to describe SSM OpsItems.
     * If the request is successful, it prints the title and status of each OpsItem.
     * If an exception occurs, it handles the error appropriately.
     */
    public void describeOpsItems(String key) {
        OpsItemFilter filter = OpsItemFilter.builder()
```

```
                .key(OpsItemFilterKey.OPS_ITEM_ID)
                .values(key)
                .operator(OpsItemFilterOperator.EQUAL)
                .build();

        DescribeOpsItemsRequest itemsRequest = DescribeOpsItemsRequest.builder()
                .maxResults(10)
                .opsItemFilters(filter)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().describeOpsItems(itemsRequest)
                    .thenAccept(itemsResponse -> {
                        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());
                        }
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeOpsItems를 참조하세요.

`DescribeParameters`

다음 코드 예시는 `DescribeParameters`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DescribeParameters를 참조하세요.

PutParameter

다음 코드 예시는 PutParameter의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 PutParameter를 참조하세요.

SendCommand

다음 코드 예시는 SendCommand의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Sends a SSM command to a managed node asynchronously.
     *
     * @param documentName The name of the document to use.
     * @param instanceId The ID of the instance to send the command to.
     * @return The command ID.
     * <p>
```

```
    * This method initiates asynchronous requests to send a SSM command to a
managed node.
    * It waits until the document is active, sends the command, and checks the
command execution status.
    */
   public String sendSSMCommand(String documentName, String instanceId) throws
InterruptedException, SsmException {
       // Before we use Document to send a command - make sure it is active.
       CompletableFuture<Void> documentActiveFuture = CompletableFuture.runAsync(()
-> {
           boolean isDocumentActive = false;
           DescribeDocumentRequest request = DescribeDocumentRequest.builder()
                   .name(documentName)
                   .build();

           while (!isDocumentActive) {
               CompletableFuture<DescribeDocumentResponse> response =
getAsyncClient().describeDocument(request);
               String documentStatus = response.join().document().statusAsString();
               if (documentStatus.equals("Active")) {
                   System.out.println("The SSM document is active and ready to
use.");
                   isDocumentActive = true;
               } else {
                   System.out.println("The SSM document is not active. Status: " +
documentStatus);
                   try {
                       Thread.sleep(5000);
                   } catch (InterruptedException e) {
                       throw new RuntimeException(e);
                   }
               }
           }
       });

       documentActiveFuture.join();

       // Create the SendCommandRequest.
       SendCommandRequest commandRequest = SendCommandRequest.builder()
               .documentName(documentName)
               .instanceIds(instanceId)
               .build();

       // Send the command.
```

```
        CompletableFuture<SendCommandResponse> commandFuture =
getAsyncClient().sendCommand(commandRequest);
        final String[] commandId = {null};

        commandFuture.whenComplete((commandResponse, ex) -> {
            if (commandResponse != null) {
                commandId[0] = commandResponse.command().commandId();
                System.out.println("Command ID: " + commandId[0]);

                // Wait for the command execution to complete.
                GetCommandInvocationRequest invocationRequest =
GetCommandInvocationRequest.builder()
                        .commandId(commandId[0])
                        .instanceId(instanceId)
                        .build();

                try {
                    System.out.println("Wait 5 secs");
                    TimeUnit.SECONDS.sleep(5);

                    // Retrieve the command execution details.
                    CompletableFuture<GetCommandInvocationResponse> invocationFuture
= getAsyncClient().getCommandInvocation(invocationRequest);
                    invocationFuture.whenComplete((commandInvocationResponse,
invocationEx) -> {
                        if (commandInvocationResponse != null) {
                            // 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);
                            }
                        } else {
                            Throwable invocationCause = (invocationEx instanceof
CompletionException) ? invocationEx.getCause() : invocationEx;
                            throw new CompletionException(invocationCause);
                        }
                    }).join();
                } catch (InterruptedException e) {
                    throw new RuntimeException(e);
                }
```

```
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof SsmException) {
                    throw (SsmException) cause;
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();

        return commandId[0];
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 SendCommand를 참조하세요.

`UpdateMaintenanceWindow`

다음 코드 예시는 `UpdateMaintenanceWindow`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    /**
     * Updates an SSM maintenance window asynchronously.
     *
     * @param id The ID of the maintenance window to update.
     * @param name The new name for the maintenance window.
     * <p>
     * This method initiates an asynchronous request to update an SSM maintenance
window.
     * If the request is successful, it prints a success message.
     * If an exception occurs, it handles the error appropriately.
     */
    public void updateSSMMaintenanceWindow(String id, String name) throws
SsmException {
```

```
        UpdateMaintenanceWindowRequest updateRequest =
UpdateMaintenanceWindowRequest.builder()
                .windowId(id)
                .allowUnassociatedTargets(true)
                .duration(24)
                .enabled(true)
                .name(name)
                .schedule("cron(0 0 ? * MON *)")
                .build();

        CompletableFuture<UpdateMaintenanceWindowResponse> future =
getAsyncClient().updateMaintenanceWindow(updateRequest);
        future.whenComplete((response, ex) -> {
            if (response != null) {
                System.out.println("The SSM maintenance window was successfully
updated");
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof SsmException) {
                    throw new CompletionException(cause);
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateMaintenanceWindow를 참조하세요.

`UpdateOpsItem`

다음 코드 예시는 `UpdateOpsItem`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   /**
    * Resolves an AWS SSM OpsItem asynchronously.
    *
    * @param opsID The ID of the OpsItem to resolve.
    * <p>
    * This method initiates an asynchronous request to resolve an SSM OpsItem.
    * If an exception occurs, it handles the error appropriately.
    */
   public void resolveOpsItem(String opsID) {
       UpdateOpsItemRequest opsItemRequest = UpdateOpsItemRequest.builder()
               .opsItemId(opsID)
               .status(OpsItemStatus.RESOLVED)
               .build();

       CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
           getAsyncClient().updateOpsItem(opsItemRequest)
                   .thenAccept(response -> {
                       System.out.println("OpsItem resolved successfully.");
                   })
                   .exceptionally(ex -> {
                       throw new CompletionException(ex);
                   }).join();
       }).exceptionally(ex -> {
           Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
           if (cause instanceof SsmException) {
               throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
           } else {
               throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
           }
       });

       try {
           future.join();
       } catch (CompletionException ex) {
           throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
       }
   }
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 UpdateOpsItem을 참조하세요.

Java 2.x용 SDK를 사용하는 Amazon Textract 예제

다음 코드 예제에서는 AWS SDK for Java 2.x Amazon Textract에서를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`AnalyzeDocument`

다음 코드 예시는 `AnalyzeDocument`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 AnalyzeDocument를 참조하세요.

`DetectDocumentText`

다음 코드 예시는 `DetectDocumentText`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

입력 문서에서 텍스트를 감지합니다.

```
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);
        }
    }
}
```

Amazon S3 버킷에 위치한 문서에서 텍스트를 감지합니다.

```
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);
        }
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 DetectDocumentText를 참조하세요.

`StartDocumentAnalysis`

다음 코드 예시는 `StartDocumentAnalysis`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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 "";
    }
}
```

- API에 대한 세부 정보는 AWS SDK for Java 2.x API 참조의 StartDocumentAnalysis를 참조하세요.

시나리오

고객 피드백 분석을 위한 애플리케이션 생성

다음 코드 예제에서는 고객 의견 카드를 분석하고, 원어에서 번역하고, 감정을 파악하고, 번역된 텍스트에서 오디오 파일을 생성하는 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

이 예제 애플리케이션은 고객 피드백 카드를 분석하고 저장합니다. 특히 뉴욕시에 있는 가상 호텔의 필요를 충족합니다. 호텔은 다양한 언어의 고객들로부터 물리적인 의견 카드의 형태로 피드백을 받습니다. 피드백은 웹 클라이언트를 통해 앱에 업로드됩니다. 의견 카드의 이미지가 업로드된 후 다음 단계가 수행됩니다.

- Amazon Textract를 사용하여 이미지에서 텍스트가 추출됩니다.
- Amazon Comprehend가 추출된 텍스트와 해당 언어의 감정을 파악합니다.
- 추출된 텍스트는 Amazon Translate를 사용하여 영어로 번역됩니다.
- Amazon Polly가 추출된 텍스트에서 오디오 파일을 합성합니다.

전체 앱은 AWS CDK를 사용하여 배포할 수 있습니다. 소스 코드와 배포 지침은 GitHub의 프로젝트를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

SDK for Java 2.x를 사용한 Amazon Transcribe 예시

다음 코드 예제에서는 Amazon Transcribe에서를 사용하여 작업을 수행하고 일반적인 시나리오를 구현 AWS SDK for Java 2.x 하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

- 시나리오

작업

`ListTranscriptionJobs`

다음 코드 예시는 `ListTranscriptionJobs`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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("--------------");
                });
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 ListTranscriptionJobs를 참조하세요.

시나리오

오디오 전사 및 작업 데이터 가져오기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- Amazon Transcribe를 통해 트랜스크립션 작업을 시작합니다.
- 작업이 완료될 때까지 기다립니다.
- 트랜스크립트가 저장되는 URI를 가져옵니다.

자세한 내용은 Amazon Transcribe 시작하기를 참조하세요.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

PCM 파일을 전사합니다.

```
/**
 * 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();
        }
    }
}
```

컴퓨터 마이크의 스트리밍 오디오를 전사합니다.

```
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);
                })
                .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 static Subscription currentSubscription;
        private final InputStream inputStream;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (currentSubscription == null) {
                currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                currentSubscription.cancel();
                currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024;
        private final Subscriber<? super AudioStream> subscriber;
```

```
        private final InputStream inputStream;
        private final ExecutorService executor = Executors.newFixedThreadPool(1);
        private final 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();
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하세요.
  - GetTranscriptionJob
  - StartTranscriptionJob

SDK for Java 2.x를 사용한 Amazon Transcribe 스트리밍 예제

다음 코드 예제에서는 Amazon Transcribe Streaming과 AWS SDK for Java 2.x 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`StartMedicalStreamTranscription`

다음 코드 예시는 `StartMedicalStreamTranscription`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/*
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

This code demonstrates the process of starting a medical transcription job using the
 AWS Transcribe
Streaming service, including setting up the audio input stream, configuring the
 transcription request,
and handling the transcription response.
 */

public class TranscribeMedicalStreamingDemoApp {
```

```
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[])
        throws ExecutionException, InterruptedException, LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
            .credentialsProvider(getCredentials())
            .build();

        CompletableFuture<Void> result =
client.startMedicalStreamTranscription(getMedicalRequest(16_000),
            new AudioStreamPublisher(getStreamFromMic()),
            getMedicalResponseHandler());

        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");
            throw new LineUnavailableException("The audio system microphone line is
not supported.");
        }

        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 StartMedicalStreamTranscriptionRequest getMedicalRequest(Integer
 mediaSampleRateHertz) {
        return StartMedicalStreamTranscriptionRequest.builder()
            .languageCode(LanguageCode.EN_US.toString()) // For medical
 transcription, EN_US is typically used.
            .mediaEncoding(MediaEncoding.PCM)
            .mediaSampleRateHertz(mediaSampleRateHertz)
            .specialty(Specialty.PRIMARYCARE) // Specify the medical specialty.
            .type(Type.CONVERSATION) // Set the type as CONVERSATION or DICTATION.
            .build();
    }

    private static StartMedicalStreamTranscriptionResponseHandler
 getMedicalResponseHandler() {
        return StartMedicalStreamTranscriptionResponseHandler.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 streamed successfully ===");
            })
            .subscriber(event -> {
                List<MedicalResult> results = ((MedicalTranscriptEvent)
 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();
        }
```

```
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartMedicalStreamTranscription을 참조하세요.

`StartStreamTranscription`

다음 코드 예시는 `StartStreamTranscription`의 사용 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
                })
```

```
                .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 static Subscription currentSubscription;
        private final InputStream inputStream;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (currentSubscription == null) {
                currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                currentSubscription.cancel();
                currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private final ExecutorService executor = Executors.newFixedThreadPool(1);
```

```
        private final 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();
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartStreamTranscription을 참조하세요.

시나리오

오디오 파일 트랜스크립션

다음 코드 예제에서는 Amazon Transcribe 스트리밍을 사용하여 소스 오디오 파일의 트랜스크립션을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * 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();
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartStreamTranscription을 참조하세요.

마이크로 오디오 트랜스크립션

다음 코드 예제에서는 Amazon Transcribe 스트리밍을 사용하여 마이크로 트랜스크립션을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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);
                })
                .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 static Subscription currentSubscription;
        private final InputStream inputStream;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (currentSubscription == null) {
                currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                currentSubscription.cancel();
                currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }
```

```
   public static class SubscriptionImpl implements Subscription {
       private static final int CHUNK_SIZE_IN_BYTES = 1024;
       private final Subscriber<? super AudioStream> subscriber;
       private final InputStream inputStream;
       private final ExecutorService executor = Executors.newFixedThreadPool(1);
       private final 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();
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 StartStreamTranscription을 참조하세요.

SDK for Java 2.x를 사용한 Amazon Translate 예제

다음 코드 예제에서는 AWS SDK for Java 2.x Amazon Translate에서를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시나리오

Amazon Lex 챗봇 구축

다음 코드 예제에서는 챗봇을 만들어 웹사이트 방문자를 참여시키는 방법을 보여줍니다.

SDK for Java 2.x

Amazon Lex API를 사용하여 웹 애플리케이션 내에 챗봇을 구축하여 웹 사이트 방문자의 참여를 유도하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

Amazon SNS 애플리케이션 구축

다음 코드 예제에서는 구독 및 게시 기능이 있고 메시지를 번역하는 애플리케이션을 만드는 방법을 보여줍니다.

SDK for Java 2.x

Amazon Simple Notification Service Java API를 사용하여 구독 및 게시 기능이 있는 웹 애플리케이션을 생성하는 방법을 보여줍니다. 또한 이 예제 애플리케이션은 메시지를 번역합니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 GitHub에서 전체 예제를 참조하세요.

Java Async API를 사용한 예제를 설정하고 실행하는 방법에 대한 전체 소스 코드와 지침은 GitHub에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon SNS
- Amazon Translate

고객 피드백 분석을 위한 애플리케이션 생성

다음 코드 예제에서는 고객 의견 카드를 분석하고, 원어에서 번역하고, 감정을 파악하고, 번역된 텍스트에서 오디오 파일을 생성하는 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Java 2.x

이 예제 애플리케이션은 고객 피드백 카드를 분석하고 저장합니다. 특히 뉴욕시에 있는 가상 호텔의 필요를 충족합니다. 호텔은 다양한 언어의 고객들로부터 물리적인 의견 카드의 형태로 피드백을 받습니다. 피드백은 웹 클라이언트를 통해 앱에 업로드됩니다. 의견 카드의 이미지가 업로드된 후 다음 단계가 수행됩니다.

- Amazon Textract를 사용하여 이미지에서 텍스트가 추출됩니다.
- Amazon Comprehend가 추출된 텍스트와 해당 언어의 감정을 파악합니다.
- 추출된 텍스트는 Amazon Translate를 사용하여 영어로 번역됩니다.
- Amazon Polly가 추출된 텍스트에서 오디오 파일을 합성합니다.

전체 앱은 AWS CDK를 사용하여 배포할 수 있습니다. 소스 코드와 배포 지침은 GitHub의 프로젝트를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

AWS SDK for Java의 버전 1.x에서 2.x로 마이그레이션

AWS SDK for Java 2.x는 Java 8+를 기반으로 구축된 1.x 코드 베이스의 대대적인 재작성본입니다. 일관성 향상, 사용 편의성, 강력한 불변성 등 많은 업데이트가 포함되어 있습니다. 이 단원에서는 버전 2.x의 새로운 주요 기능을 설명하고 코드를 1.x에서 버전 2.x로 마이그레이션하는 방법에 대한 지침을 제공합니다.

주제

버전 2의 새 기능

- 고유한 HTTP 클라이언트를 구성할 수 있습니다. HTTP 전송 구성을 참조하세요.
- 비동기식 클라이언트에는 비차단 I/O 지원 및 `CompletableFuture` 객체 반환 기능이 있습니다. 비동기 프로그래밍을 참조하세요.
- 여러 페이지를 반환하는 작업에는 자동으로 페이지가 매겨진 응답이 있습니다. 이렇게 하면 후속 페이지를 찾아 가져올 필요 없이 응답으로 수행할 작업에 대한 코드를 집중할 수 있습니다. 페이지 매김을 참조하세요.
- AWS Lambda 함수의 SDK 시작 시간 성능이 개선되었습니다. SDK 시작 시간 성능 개선을 참조하세요.
- 버전 2.x는 요청을 생성하는 새로운 속기 방법을 지원합니다.

  Example

  ```
  dynamoDbClient.putItem(request -> request.tableName(TABLE))
  ```

새 기능에 대한 자세한 내용과 특정 코드 예제를 보려면 이 가이드의 다른 단원을 참조하세요.

- 빠른 시작

- 설정
- AWS SDK for Java 2.x용 코드 예제
- SDK 사용
- AWS SDK for Java을 위한 보안

코드를 AWS SDK for Java 1.x에서 2.x로 마이그레이션하는 방법

몇 가지 방법으로 기존 SDK for Java 1.x 애플리케이션을 마이그레이션할 수 있습니다.

1. 마이그레이션 도구를 사용한 자동 접근 방식
2. 1.x 가져오기를 2.x 가져오기로 점진적으로 대체하는 수동 접근 방식입니다.

먼저 마이그레이션 도구를 사용하는 것이 좋습니다. 1.x에서 2.x 코드로 일상적인 대체 작업의 대부분을 자동화합니다.

도구로 모든 기능을 마이그레이션할 수 없으므로 도구를 실행한 후 나머지 v1 코드를 검색해야 합니다. 도구를 통해 마이그레이션되지 않은 코드를 찾으면 단계별 설명(수동 접근 방식)을 따르고 마이그레이션 가이드 문서를 사용하여 마이그레이션을 완료합니다.

주제

- AWS SDK for Java 마이그레이션 도구
- 마이그레이션에 대한 단계별 설명(예제 포함)

AWS SDK for Java 마이그레이션 도구

AWS SDK for Java는 SDK for Java 1.x(V1) 코드를 2.x(V2)로 마이그레이션하는 작업을 자동화하는 데 도움이 되는 마이그레이션 도구를 제공합니다. 이 도구는 오픈 소스, 소스 코드 리팩터링 도구인 OpenRewrite를 사용하여 마이그레이션을 수행합니다. OpenRewrite는 코드 수정 규칙(다른 명칭: '레시피')을 사용하여 소스 코드를 V1에서 V2 구문 및 패턴으로 자동으로 업데이트합니다.

도구는 SDK 서비스 클라이언트 및 S3 Transfer Manager 상위 수준 라이브러리에 대한 코드 수정 규칙을 지원합니다. V1의 `DynamoDBMapper`에서 V2로의 DynamoDB 향상된 클라이언트 API와 같은 다른 상위 수준 API에 대한 코드 수정 규칙은 지원되지 않습니다.

제한 사항에 대한 자세한 내용은 이 페이지의 마지막 부분을 참조하세요. 수동 마이그레이션 단계에서 지원되지 않는 일반적인 코드 패턴의 자세한 예제는 지원되지 않는 코드 패턴을 참조하세요.

마이그레이션 도구 사용

Maven 프로젝트 마이그레이션

아래 설명에 따라 OpenRewrite Maven 플러그인 도구를 사용하여 SDK for Java 1.x Maven 기반 프로젝트를 마이그레이션합니다.

1. Maven 프로젝트의 루트 디렉터리로 이동

   터미널(명령줄) 창을 열고 Maven 기반 애플리케이션의 루트 디렉터리로 이동합니다.

2. 플러그인의 `rewrite-maven-plugin` 명령 실행

   `dryRun` 및 `run`의 2가지 모드(Maven 목표) 중에서 선택할 수 있습니다.

   `dryRun` 모드

   `dryRun` 모드에서 플러그인은 콘솔 출력에 diff 로그를 생성하고 `target/rewrite` 폴더에 `rewrite.patch`라는 패치 파일을 생성합니다. 이 모드를 사용하면 소스 코드 파일을 변경하지 않으므로 변경할 내용을 미리 볼 수 있습니다.

   다음 예제에서는 `dryRun` 모드에서 플러그인을 호출하는 방법을 보여줍니다.

```
mvn org.openrewrite.maven:rewrite-maven-plugin:<rewrite-plugin-version>*:dryRun \
  -Drewrite.recipeArtifactCoordinates=software.amazon.awssdk:v2-
migration:<sdkversion>** \
  -Drewrite.activeRecipes=software.amazon.awssdk.v2migration.AwsSdkJavaV1ToV2
```

   **<rewrite-plugin-version>*을 이 테스트 파일에 표시된 `rewriteMavenPluginVersion` 값으로 교체합니다.

   ***<sdkversion>*을 2.x SDK 버전으로 교체합니다. Maven Central에서 최신 버전을 확인하세요.

> **Important**
>
> 다른 버전이 작동하지 않을 수 있으므로 테스트 파일에 표시된 버전의 `rewrite-maven-plugin`을 사용해야 합니다.

   `dryRun` 모드의 콘솔 출력은 다음 출력과 유사해야 합니다.

```
[WARNING] These recipes would make changes to project/src/test/resources/maven/
before/pom.xml:
[WARNING]     software.amazon.awssdk.v2migration.AwsSdkJavaV1ToV2
[WARNING]         software.amazon.awssdk.v2migration.UpgradeSdkDependencies
[WARNING]             org.openrewrite.java.dependencies.AddDependency:
 {groupId=software.amazon.awssdk, artifactId=apache-client, version=2.27.0,
 onlyIfUsing=com.amazonaws.ClientConfiguration}
[WARNING]             org.openrewrite.java.dependencies.AddDependency:
 {groupId=software.amazon.awssdk, artifactId=netty-nio-client, version=2.27.0,
 onlyIfUsing=com.amazonaws.ClientConfiguration}
[WARNING]             org.openrewrite.java.dependencies.ChangeDependency:
 {oldGroupId=com.amazonaws, oldArtifactId=aws-java-sdk-bom,
 newGroupId=software.amazon.awssdk, newArtifactId=bom, newVersion=2.27.0}
[WARNING]             org.openrewrite.java.dependencies.ChangeDependency:
 {oldGroupId=com.amazonaws, oldArtifactId=aws-java-sdk-s3,
 newGroupId=software.amazon.awssdk, newArtifactId=s3, newVersion=2.27.0}
[WARNING]             org.openrewrite.java.dependencies.ChangeDependency:
 {oldGroupId=com.amazonaws, oldArtifactId=aws-java-sdk-sqs,
 newGroupId=software.amazon.awssdk, newArtifactId=sqs, newVersion=2.27.0}
[WARNING] These recipes would make changes to project/src/test/resources/maven/
before/src/main/java/foo/bar/Application.java:
[WARNING]     software.amazon.awssdk.v2migration.AwsSdkJavaV1ToV2
[WARNING]         software.amazon.awssdk.v2migration.S3GetObjectConstructorToFluent
[WARNING]             software.amazon.awssdk.v2migration.ConstructorToFluent
[WARNING]         software.amazon.awssdk.v2migration.S3StreamingResponseToV2
[WARNING]         software.amazon.awssdk.v2migration.ChangeSdkType
[WARNING]         software.amazon.awssdk.v2migration.ChangeSdkCoreTypes
[WARNING]             software.amazon.awssdk.v2migration.ChangeExceptionTypes
[WARNING]                 org.openrewrite.java.ChangeType:
 {oldFullyQualifiedTypeName=com.amazonaws.AmazonClientException,
 newFullyQualifiedTypeName=software.amazon.awssdk.core.exception.SdkException}
[WARNING]                 org.openrewrite.java.ChangeMethodName:
 {methodPattern=com.amazonaws.AmazonServiceException getRequestId(),
 newMethodName=requestId}
[WARNING]                 org.openrewrite.java.ChangeMethodName:
 {methodPattern=com.amazonaws.AmazonServiceException getErrorCode(),
 newMethodName=awsErrorDetails().errorCode}
[WARNING]                 org.openrewrite.java.ChangeMethodName:
 {methodPattern=com.amazonaws.AmazonServiceException getServiceName(),
 newMethodName=awsErrorDetails().serviceName}
[WARNING]                 org.openrewrite.java.ChangeMethodName:
 {methodPattern=com.amazonaws.AmazonServiceException getErrorMessage(),
 newMethodName=awsErrorDetails().errorMessage}
```

```
[WARNING]                org.openrewrite.java.ChangeMethodName:
 {methodPattern=com.amazonaws.AmazonServiceException getRawResponse(),
 newMethodName=awsErrorDetails().rawResponse().asByteArray}
[WARNING]                org.openrewrite.java.ChangeMethodName:
 {methodPattern=com.amazonaws.AmazonServiceException getRawResponseContent(),
 newMethodName=awsErrorDetails().rawResponse().asUtf8String}
[WARNING]                org.openrewrite.java.ChangeType:
 {oldFullyQualifiedTypeName=com.amazonaws.AmazonServiceException,
 newFullyQualifiedTypeName=software.amazon.awssdk.awscore.exception.AwsServiceException}
[WARNING]         software.amazon.awssdk.v2migration.NewClassToBuilderPattern
[WARNING]             software.amazon.awssdk.v2migration.NewClassToBuilder
[WARNING]             software.amazon.awssdk.v2migration.V1SetterToV2
[WARNING]         software.amazon.awssdk.v2migration.V1GetterToV2
...
[WARNING]         software.amazon.awssdk.v2migration.V1BuilderVariationsToV2Builder
[WARNING]         software.amazon.awssdk.v2migration.NewClassToBuilderPattern
[WARNING]             software.amazon.awssdk.v2migration.NewClassToBuilder
[WARNING]             software.amazon.awssdk.v2migration.V1SetterToV2
[WARNING]         software.amazon.awssdk.v2migration.HttpSettingsToHttpClient
[WARNING]         software.amazon.awssdk.v2migration.WrapSdkClientBuilderRegionStr
[WARNING] Patch file available:
[WARNING]     project/src/test/resources/maven/before/target/rewrite/rewrite.patch
[WARNING] Estimate time saved: 20m
[WARNING] Run 'mvn rewrite:run' to apply the recipes.
```

`run` 모드

플러그인을 `run` 모드에서 실행할 경우 디스크의 소스 코드를 수정하여 변경 사항을 적용합니다. 명령을 실행하기 전에 소스 코드의 백업이 있는지 확인합니다.

다음 예제에서는 `run` 모드에서 플러그인을 호출하는 방법을 보여줍니다.

```
mvn org.openrewrite.maven:rewrite-maven-plugin:<rewrite-plugin-version>*:run \
  -Drewrite.recipeArtifactCoordinates=software.amazon.awssdk:v2-
migration:<sdkversion>** \
  -Drewrite.activeRecipes=software.amazon.awssdk.v2migration.AwsSdkJavaV1ToV2
```

**<rewrite-plugin-version>*</rewrite-plugin-version>*을 이 테스트 파일에 표시된 `rewriteMavenPluginVersion`value로 교체합니다.

***<sdkversion>*</sdkversion>*을 2.x SDK 버전으로 교체합니다. Maven Central에서 최신 버전을 확인하세요.

명령을 실행한 후 애플리케이션을 컴파일하고 테스트를 실행하여 변경 사항을 확인합니다.

Gradle 프로젝트 마이그레이션

아래 설명에 따라 OpenRewrite Gradle 플러그인 도구를 사용하여 SDK for Java 1.x Gradle 기반 프로젝트를 마이그레이션합니다.

1. Gradle 프로젝트의 루트 디렉터리로 이동

   터미널(명령줄) 창을 열고 Gradle 기반 애플리케이션의 루트 디렉터리로 이동합니다.

2. Gradle init 스크립트 만들기

   디렉터리에 다음 콘텐츠로 `init.gradle`이라는 파일을 만듭니다.

```
initscript {
    repositories {
        maven { url "https://plugins.gradle.org/m2" }
    }
    dependencies {
        classpath("org.openrewrite:plugin:<rewrite-plugin-version>*")
    }
}

rootProject {
    plugins.apply(org.openrewrite.gradle.RewritePlugin)
    dependencies {
        rewrite("software.amazon.awssdk:v2-migration:latest.release")
    }

    afterEvaluate {
        if (repositories.isEmpty()) {
            repositories {
                mavenCentral()
            }
        }
    }
}
```

   **<rewrite-plugin-version>*을 이 테스트 파일에 표시된 버전으로 교체합니다.

3. `rewrite` 명령 실행

Maven 플러그인과 마찬가지로 dryRun 또는 run 모드에서 Gradle 플러그인을 실행할 수 있습니다.

dryRun 모드

다음 예제에서는 dryRun 모드에서 플러그인을 호출하는 방법을 보여줍니다.

```
gradle rewriteDryRun --init-script init.gradle \
  -Drewrite.activeRecipes=software.amazon.awssdk.v2migration.AwsSdkJavaV1ToV2
```

run 모드

다음 예제에서는 run 모드에서 플러그인을 호출하는 방법을 보여줍니다.

```
gradle rewriteRun --init-script init.gradle \
  -Drewrite.activeRecipes=software.amazon.awssdk.v2migration.AwsSdkJavaV1ToV2
```

현재 제한 사항

마이그레이션은 V2와 동일하게 업데이트되는 코드 수정 규칙을 통해 대부분의 V1 코드를 지원하지만, 일부 클래스와 메서드는 적용되지 않습니다. 이러한 클래스 및 메서드의 경우 단계별 설명에 따라 코드를 수동으로 마이그레이션합니다.

일부 지원되지 않는 코드 수정 규칙의 경우 마이그레이션 도구는 다음으로 시작하는 주석을 추가할 수 있습니다.

```
/*AWS SDK for Java v2 migration: Transform for ...
```

주석 다음에 도구는 메서드 또는 클래스의 V2 버전의 일반 스터브를 출력합니다. 예를 들어 다음 출력에서 마이그레이션 도구는 V1 S3 클라이언트의 setBucketLifecycleConfiguration 메서드를 마이그레이션하려고 시도했습니다.

```
/*AWS SDK for Java v2 migration: Transform for setBucketLifecycleConfiguration method
 not supported.
Please manually migrate your code by using builder pattern, update from
 BucketLifecycleConfiguration.Rule
to LifecycleRule, StorageClass to TransitionStorageClass, and adjust imports and
 names.*/
s3.putBucketLifecycleConfiguration(
        PutBucketLifecycleConfigurationRequest.builder()
```

```
            .bucket(bucketName)
            .lifecycleConfiguration(BucketLifecycleConfiguration.builder()
                .build())
            .build());
```

아래 목록의 링크를 클릭하면 코드를 수동으로 마이그레이션하는 데 도움이 되는 마이그레이션 정보로 이동합니다.

- the section called “S3 클라이언트 차이”
- S3 Transfer Manager(TransferManager)
- DynamoDB 객체 매핑(DynamoDBMapper)
- EC2 메타데이터 유틸리티(EC2MetadataUtils)
- 웨이터(AmazonDynamoDBWaiters)
- IAM 정책 빌더(정책)
- CloudFront 사전 서명(CloudFrontUrlSigner, CloudFrontCookieSigner)
- S3 이벤트 알림(S3EventNotification)
- SDK 지표 게시(1.x 설명서, 2.x 설명서)
- 지원되지 않는 코드 패턴 - 수동 마이그레이션이 필요한 일반적인 코드 패턴의 세부 예제

마이그레이션 도구의 지원되지 않는 코드 패턴

마이그레이션 도구는 대부분의 v1 코드를 v2로 자동 변환합니다. 그러나 일부 코드 패턴은 수동 마이그레이션이 필요합니다. 이 주제에서는 지원되지 않는 가장 일반적인 패턴의 세부 예제를 제공하고 이를 수동으로 변환하는 방법을 보여줍니다.

다음 패턴 목록이 전부는 아닙니다. 마이그레이션 도구를 실행한 후에도 코드가 컴파일되지 않으면 단계별 마이그레이션 설명에 따라 나머지 v1 코드를 수동으로 마이그레이션합니다.

파라미터를 사용하여 객체 생성자 요청

요청 POJO(Amazon S3 제외)의 경우 마이그레이션 도구는 setter 메서드만 변환합니다. 이 도구는 파라미터가 있는 생성자를 지원하지 않습니다.

지원되는 패턴: setter를 사용하여 객체 요청(작성자 파라미터 없음)

이전(원래 v1 코드):

```
import com.amazonaws.services.sqs.model.SendMessageRequest;

SendMessageRequest request = new SendMessageRequest().withMessageBody("Hello World");
request.setQueueUrl("https://sqs.us-west-2.amazonaws.com/0123456789012/demo-queue");
```

이후(마이그레이션 도구 결과):

```
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;

SendMessageRequest request = SendMessageRequest.builder()
    .messageBody("Hello World").build();
request = request.toBuilder()
    .queueUrl("https://sqs.us-west-2.amazonaws.com/0123456789012/demo-queue").build();
```

지원되지 않는 패턴: 파라미터를 사용하여 객체 생성자 요청

마이그레이션 도구는 파라미터가 있는 생성자를 변환할 수 없습니다.

수동 마이그레이션 전, 마이그레이션 도구 사용 후:

```
import software.amazon.awssdk.services.sqs.model.SendMessageRequest; // Import updated
 to v2.

// This pattern requires manual migration.
SendMessageRequest request = new SendMessageRequest(
    "https://sqs.us-west-2.amazonaws.com/0123456789012/demo-queue",
    "Hello World");
```

마이그레이션 도구는 가져오기를 v2로 변환하지만 생성자 코드는 변경되지 않고 빌더 패턴을 사용하려면 수동 업데이트가 필요합니다.

수동 마이그레이션 후:

```
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;

SendMessageRequest request = SendMessageRequest.builder()
    .messageBody("Hello World")
    .queueUrl("https://sqs.us-west-2.amazonaws.com/0123456789012/demo-queue")
    .build();
```

개별 파라미터가 있는 서비스 클라이언트 메서드

마이그레이션 도구는 요청 객체(Amazon S3 제외) 대신 개별 파라미터를 사용하는 서비스 클라이언트 메서드를 변환할 수 없습니다.

이전(v1 코드):

```
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClient;

AmazonSQS sqs = new AmazonSQSClient();
// The following v1 method takes individual parameters.
sqs.sendMessage("https://sqs.us-west-2.amazonaws.com/0123456789012/demo-queue", "Hello
 World");
```

이후(마이그레이션 도구 결과 - 컴파일되지 않음):

```
import software.amazon.awssdk.services.sqs.SqsClient;  // Import updated to v2.
// No import statement for the v2 request POJO.

SqsClient sqs = SqsClient.builder().build();

// Does not compile-v2 methods only accept request POJOs.
sqs.sendMessage("https://sqs.us-west-2.amazonaws.com/0123456789012/demo-queue", "Hello
 World");
```

요청 객체를 사용하려면 메서드 인수를 수동으로 업데이트해야 합니다.

```
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest; // Add manually.

SqsClient sqs = SqsClient.builder().build();

// Corrected v2 code.
SendMessageRequest request = SendMessageRequest.builder()
    .queueUrl("https://sqs.us-west-2.amazonaws.com/0123456789012/demo-queue")
    .messageBody("Hello World")
    .build();
sqs.sendMessage(request);
```

요청 제한 시간 메서드

마이그레이션 도구는 요청 객체에 제한 시간을 설정하는 메서드를 변환하지 않습니다.

이전(v1 코드):

```
import com.amazonaws.services.sqs.model.SendMessageRequest;

SendMessageRequest request = new SendMessageRequest();
request.setSdkRequestTimeout(7);
```

이후(마이그레이션 도구 결과 - 컴파일되지 않음):

```
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;  // Import updated
 to v2.

SendMessageRequest request = SendMessageRequest.builder().build();

// Does not compile.
request.setSdkRequestTimeout(7);
```

v2의 `overrideConfiguration` 메서드를 사용하려면 수동으로 마이그레이션해야 합니다.

```
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
import java.time.Duration;

SendMessageRequest request = SendMessageRequest.builder().build();

// Corrected v2 code.
request = request.toBuilder()
    .overrideConfiguration(o -> o.apiCallTimeout(Duration.ofSeconds(7)))
    .build();
```

파라미터가 있는 서비스 클라이언트 생성자

마이그레이션 도구는 비어 있는 서비스 클라이언트 생성자를 변환하지만 자격 증명 또는 구성과 같은 파라미터를 허용하는 생성자는 변환할 수 없습니다.

이전(v1 코드):

```
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClient;
```

```
AWSCredentials awsCredentials = new BasicAWSCredentials("akid", "skid");
AmazonSQS sqs = new AmazonSQSClient(awsCredentials);
```

이후(마이그레이션 도구 결과 - 컴파일되지 않음):

```
import software.amazon.awssdk.auth.credentials.AwsCredentials;
import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;
import software.amazon.awssdk.services.sqs.SqsClient;  // Import updated to v2.

AwsCredentials awsCredentials = AwsBasicCredentials.create("akid", "skid");

// Does not compile.
SqsClient sqs = new SqsClient(awsCredentials);
```

빌더 패턴을 사용하려면 서비스 클라이언트 생성자를 수동으로 업데이트해야 합니다.

```
import software.amazon.awssdk.auth.credentials.AwsCredentials;
import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;
import software.amazon.awssdk.auth.credentials.StaticCredentialsProvider;  // Add
 manually.
import software.amazon.awssdk.services.sqs.SqsClient;

AwsCredentials awsCredentials = AwsBasicCredentials.create("akid", "skid");

// Corrected v2 code.
SqsClient sqs = SqsClient.builder()
    .credentialsProvider(StaticCredentialsProvider.create(awsCredentials))
    .build();
```

마이그레이션에 대한 단계별 설명(예제 포함)

이 섹션에서는 현재 SDK for Java v1.x를 사용하는 애플리케이션을 SDK for Java 2.x로 마이그레이션하기 위한 단계별 안내서를 제공합니다. 첫 번째 부분에서는 단계에 대한 개요와 마이그레이션의 세부 예제를 제공합니다.

여기에서 다루는 단계는 애플리케이션이 모델 기반 서비스 클라이언트를 사용하여 AWS 서비스를 호출하는 일반적인 사용 사례의 마이그레이션을 설명합니다. S3 Transfer Manager 또는 CloudFront 사전 서명과 같은 개괄적인 API를 사용하는 코드를 마이그레이션해야 하는 경우 the section called “1.x와 2.x의 차이점” 목차 아래의 섹션을 참조하세요.

여기에 설명된 접근 방식은 제안입니다. 다른 기법을 사용하고 IDE의 코드 편집 기능을 활용하여 동일한 결과를 얻을 수 있습니다.

단계 개요

1. 먼저 SDK for Java 2.x BOM을 추가합니다.

SDK for Java 2.x의 Maven BOM(Bill of Materials) 요소를 POM 파일에 추가하면 필요한 모든 v2 종속성이 동일한 버전인지 확인할 수 있습니다. POM에는 v1 및 v2 종속성이 모두 포함될 수 있습니다. 이렇게 하면 코드를 한 번에 모두 변경하는 대신 점진적으로 마이그레이션할 수 있습니다.

SDK for Java 2.x BOM

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>bom</artifactId>
      <version>2.27.21</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

Maven 중앙 리포지토리에서 최신 버전을 확인해 보세요.

2. v1 클래스 가져오기 문에 대한 파일 검색

애플리케이션의 파일에서 v1 가져오기에 사용되는 SERVICE_ID를 스캔하면 사용된 고유한 SERVICE_ID를 확인할 수 있습니다. SERVICE_ID는 AWS 서비스의 짧고 고유한 이름입니다. 예를 들어 `cognitoidentity`는 Amazon Cognito 자격 증명의 SERVICE_ID입니다.

3. v1 가져오기 문에서 v2 Maven 종속성 확인

모든 고유한 v1 SERVICE_ID를 찾은 후 the section called “artifactId 매핑에 대한 패키지 이름”을 참조하여 v2 종속성에 해당하는 Maven 아티팩트를 확인할 수 있습니다.

4. POM 파일에 v2 종속성 요소 추가

3단계에서 확인한 종속성 요소로 Maven POM 파일을 업데이트합니다.

5. Java 파일에서 v1 클래스를 v2 클래스로 점진적으로 변경합니다.

v1 클래스를 v2 클래스로 대체할 때 생성자 대신 빌더를 사용하고 유용한 getter 및 setter를 사용하는 등 v2 API를 지원하는 데 필요한 변경 사항을 적용합니다.

6. POM에서 v1 Maven 종속성 제거 및 파일에서 v1 가져오기

v2 클래스를 사용하도록 코드를 마이그레이션한 후 파일에서 남은 v1 가져오기와 빌드 파일에서 모든 종속성을 제거합니다.

7. v2 API 개선 사항을 사용하도록 코드 리팩터링

코드가 성공적으로 컴파일되고 테스트를 통과하면 다른 HTTP 클라이언트 또는 페이지 매김 도구를 사용하여 코드를 단순화하는 등 v2 개선 사항을 활용할 수 있습니다. 이 단계는 선택 사항입니다.

마이그레이션 예제

이 예제에서는 SDK for Java v1을 사용하고 여러 AWS 서비스에 액세스하는 애플리케이션을 마이그레이션합니다. 5단계에서 다음 v1 메서드를 자세히 살펴봅니다. 이는 8개의 메서드가 포함된 클래스의 한 가지 메서드이며 애플리케이션에는 32개의 클래스가 있습니다.

마이그레이션할 v1 메서드

Java 파일에서 v1 SDK 가져오기만 아래에 나열됩니다.

```
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. v2 Maven BOM 추가

dependencyManagement 섹션의 다른 종속성과 함께 SDK for Java 2.x용 Maven BOM을 POM에 추가합니다. POM 파일에 SDK v1용 BOM이 있는 경우 지금은 그대로 둡니다. 이후 단계에서 제거됩니다.

시작 시 POM 종속성 관리

```
<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>2.27.21</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

2. v1 클래스 가져오기 문에 대한 파일 검색

애플리케이션의 코드를 검색하여 `import com.amazonaws.services`의 고유한 발생을 찾습니다. 이를 통해 프로젝트에서 사용하는 v1 종속성을 확인할 수 있습니다. 애플리케이션에 v1 종속성이 나열된 Maven POM 파일이 있는 경우 이 정보를 대신 사용할 수 있습니다.

이 예제에서는 [ripgrep(rg)](#) 명령을 사용하여 코드 베이스를 검색합니다.

코드 베이스의 루트에서 다음 `ripgrep` 명령을 실행합니다. `ripgrep`가 가져오기 문을 찾은 후 `cut`, `sort` 및 `uniq` 명령으로 파이프 처리되어 SERVICE_ID를 격리합니다.

```
rg --no-filename 'import\s+com\.amazonaws\.services' | cut -d '.' -f 4 | sort | uniq
```

이 애플리케이션의 경우 다음 SERVICE_ID가 콘솔에 로그됩니다.

```
autoscaling
cloudformation
ec2
```

```
identitymanagement
```

이는 `import` 문에 사용된 다음 패키지 이름 각각이 한 번 이상 발생했음을 나타냅니다. 이 목적에 따라 개별 클래스 이름은 중요하지 않습니다. 사용되는 SERVICE_ID를 찾기만 하면 됩니다.

```
com.amazonaws.services.autoscaling.*
com.amazonaws.services.cloudformation.*
com.amazonaws.services.ec2.*
com.amazonaws.services.identitymanagement.*
```

3. v1 가져오기 문에서 v2 Maven 종속성 확인

2단계에서 격리한 v1용 SERVICE_ID를 예로 들면 `autoscaling` 및 `cloudformation`은 대부분 동일한 v2 SERVICE_ID에 매핑할 수 있습니다. v2 Maven artifactId는 대부분의 경우 SERVICE_ID와 일치하므로 POM 파일에 종속성 블록을 추가하는 데 필요한 정보가 포함됩니다.

다음 테이블은 v2 종속성을 확인하는 방법을 보여줍니다.

v1 SERVICE_ID는 ...에 매핑됩니다. 패키지 이름	v2 SERVICE_ID는 ...에 매핑됩니다. 패키지 이름	v2 Maven 종속성

EC2
 	
`com.amazonaws.services. ec2.*`
	EC2
	
	`software.amazon.awssdk.services. ec2.*`
		`<dependency>`
		`  <groupId>software.amazon.awssdk</groupId>`
		`  <artifactId> ec2</artifactId>`
		`</dependency>`

Auto Scaling
 	
`com.amazonaws.services. autoscaling .*`
	Auto Scaling
	
	`software.amazon.awssdk.services. autoscaling .*`
		`<dependency>`
		`  <groupId>software.amazon.awssdk</groupId>`
		`  <artifactId> autoscaling </artifactId>`
		`</dependency>`

cloudformation/	cloudformation/	`<dependency>`

v1 SERVICE_ID는 ...에 매핑됩니다. 패키지 이름	v2 SERVICE_ID는 ...에 매핑됩니다. 패키지 이름	v2 Maven 종속성
`com.amazonaws.services. cloudformation .*`	`software.amazon.awssdk. cloudformation .*`	`<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>`

* `iam` 매핑에 대한 `identitymanagement`는 SERVICE_ID가 버전 간에 다른 예외입니다. Maven 또는 Gradle이 v2 종속성을 해결할 수 없는 경우 the section called “artifactId 매핑에 대한 패키지 이름” 예외는 섹션을 참조하세요.

4. POM 파일에 v2 종속성 요소 추가

3단계에서는 POM 파일에 추가해야 하는 4개의 종속성 블록을 확인했습니다. 1단계에서 BOM을 지정했으므로 버전을 추가할 필요가 없습니다. 가져오기가 추가되면 POM 파일에 다음과 같은 종속성 요소가 포함됩니다.

```
  ...
<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. Java 파일에서 v1 클래스를 v2 클래스로 점진적으로 변경합니다.

마이그레이션하는 메서드에서 다음을 확인합니다.

- `com.amazonaws.services.ec2.AmazonEC2Client`의 EC2 서비스 클라이언트입니다.
- 여러 EC2 모델 클래스가 사용됩니다. 예: `DescribeInstancesRequest` 및 `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;
}
...
```

목표는 모든 v1 가져오기를 v2 가져오기로 대체하는 것입니다. 한 번에 하나의 클래스를 진행합니다.

a. 가져오기 문 또는 클래스 이름 교체

`describeRunningInstances` 메서드의 첫 번째 파라미터는 v1 `AmazonEC2Client` 인스턴스입니다. 다음 중 하나를 수행합니다.

- `com.amazonaws.services.ec2.AmazonEC2Client`에 대한 가져오기를 `software.amazon.awssdk.services.ec2.Ec2Client`로 대체하고 `AmazonEC2Client`를 `Ec2Client`로 변경합니다.
- 파라미터 유형을 `Ec2Client`로 변경하고 IDE가 올바른 가져오기를 표시하도록 합니다. 클라이언트 이름이 `AmazonEC2Client` 및 `Ec2Client`와 다르기 때문에 IDE는 v2 클래스를 가져오라는 메시지를 표시합니다. 클래스 이름이 두 버전에서 동일한 경우에는 이 접근 방식이 작동하지 않습니다.

b. v1 모델 클래스를 v2와 동등한 클래스로 교체

v2 `Ec2Client`를 변경한 후 IDE를 사용하면 다음 문에 컴파일 오류가 표시됩니다.

```
result = ec2.describeInstances(request);
```

컴파일 오류는 v1의 `DescribeInstancesRequest` 인스턴스를 v2 `Ec2Client` `describeInstances` 메서드의 파라미터로 사용하여 발생합니다. 수정하려면 다음의 대체 또는 가져오기 문을 만듭니다.

replace	여기서 DCG와 iDCG는 각각 다음과 같습니다.
`import com.amazonaws.services.ec2.model.DescribeInstancesRequest`	`import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest`

c. v1 생성자를 v2 빌더로 변경합니다.

v2 클래스에 생성자가 없기 때문에 컴파일 오류가 계속 표시됩니다. 수정하려면 다음 변경 사항을 적용합니다.

변경	아래로 변경합니다.

`final DescribeInstancesRequest request = new DescribeInstancesRequest()`
`.withInstanceIds(instanceIdsCopy);`
	`final DescribeInstancesRequest request = DescribeInstancesRequest.builder()`
	`.instanceIds(instanceIdsCopy)`
	`.build();`

d. v1 **`*Result`** 응답 객체를 v2 **`*Response`**에 상응하는 객체로 교체

v1과 v2의 일관된 차이점은 v2의 모든 응답 객체가 `*Result` 대신 `*Response`로 끝나는 것입니다. v1 `DescribeInstancesResult` 가져오기를 v2 가져오기인 `DescribeInstancesResponse`로 교체합니다.

d. API 변경 사항 적용

다음 문은 몇 가지 변경 사항이 필요합니다.

```
request.setNextToken(result.getNextToken());
```

v2에서 setter 메서드는 `set`를 사용하거나 `prefix`와 함께 사용하지 않습니다. 또한 접두사가 `get`인 getter 메서드는 SDK for Java 2.x에서 삭제됩니다.

`request` 인스턴스 등의 모델 클래스는 v2에서 변경할 수 없으므로 빌더로 새 `DescribeInstancesRequest`를 만들어야 합니다.

v2에서 문은 다음과 같습니다.

```
request = DescribeInstancesRequest.builder()
        .nextToken(result.nextToken())
        .build();
```

d. 메서드가 v2 클래스로 컴파일될 때까지 반복합니다.

나머지 코드를 계속 진행합니다. v1 가져오기를 v2 가져오기로 교체하고 컴파일 오류를 수정합니다. v2 API 참조 및 필요에 따라 다른 참조를 참고합니다.

이 단일 메서드를 마이그레이션하면 다음의 v2 코드가 포함됩니다.

```
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;
    }
...
```

8가지 메서드로 Java 파일의 단일 메서드를 마이그레이션하기 때문에 파일을 작업할 때 v1과 v2 가져오기가 혼합되어 있습니다. 단계를 수행할 때 마지막 6개의 가져오기 문을 추가했습니다.

모든 코드를 마이그레이션하면 더 이상 v1 가져오기 문이 없습니다.

6. POM에서 v1 Maven 종속성 제거 및 파일에서 v1 가져오기

파일의 모든 v1 코드를 마이그레이션하면 다음과 같은 v2 SDK 가져오기 문이 있습니다.

```
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;
```

애플리케이션의 모든 파일을 마이그레이션하면 더 이상 POM 파일에 v1 종속성이 필요하지 않습니다. 모든 v1 종속성 블록을 사용하는 경우 dependencyManagement 섹션에서 v1 BOM을 제거합니다.

7. v2 API 개선 사항을 사용하도록 코드 리팩터링

마이그레이션 중인 코드 조각의 경우 v2 페이지 매김 도구를 필요에 따라 사용하고 SDK가 더 많은 데이터에 대한 토큰 기반 요청을 관리하도록 할 수 있습니다.

전체 do절을 다음으로 교체할 수 있습니다.

```
               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);
                                   }
                               }));
```

Maven artifactId 매핑에 대한 패키지 이름

Maven 또는 Gradle 프로젝트를 SDK for Java의 v1에서 v2로 마이그레이션할 때 빌드 파일에 추가할 종속성을 파악해야 합니다. the section called “단계별 설명”(3단계)에 설명된 접근 방식은 가져오기 문의 패키지 이름을 시작점으로 사용하여 빌드 파일에 추가할 종속성(artifactId)을 결정합니다.

이 주제의 정보를 사용하여 v1 패키지 이름을 v2 artifactId로 매핑합니다.

패키지 이름 및 Maven artifactId에 사용되는 일반적인 이름 지정 규칙

다음 테이블에는 SDK가 주어진 SERVICE_ID에 사용하는 일반적인 이름 지정 규칙이 나와 있습니다. SERVICE_ID는 AWS 서비스의 고유 식별자입니다. 예를 들어 Amazon S3 서비스의 SERVICE_ID는 `s3`이고 `cognitoidentity`는 Amazon Cognito 자격 증명의 SERVICE_ID입니다.

v1 패키지 이름(가져오기 문)	v1 artifactId	v2 artifactId	v2 패키지 이름(가져오기 문)
com.amazonaws.services.SERVICE_ID	aws-java-sdk-SERVICE_ID	SERVICE_ID	software.amazon.awssdk.services.SERVICE_ID
Amazon Cognito 자격 증명의 예(SERVICE_ID: ***`cognitoidentity`***)			
com.amazonaws.services.cognitoidentity	aws-java-sdk-cognitoidentity	cognitoidentity	software.amazon.awssdk.services.cognitoidentity

SERVICE_ID 차이점

v1 내

경우에 따라 SERVICE_ID는 패키지 이름과 동일한 서비스의 artifactId 간에 다릅니다. 예를 들어 다음 테이블의 CloudWatch Metrics 행은 `metrics`이 패키지 이름의 SERVICE_ID이지만 `cloudwatchmetrics`은 artifactId의 SERVICE_ID임을 보여줍니다.

v2 내

패키지 이름 및 artifactId에 사용되는 SERVICE_ID에는 차이가 없습니다.

v1 및 v2 차이

대부분의 서비스에서 v2의 SERVICE_ID는 패키지 이름과 artifactId 모두에서 v1의 SERVICE_ID와 동일합니다. 이전 테이블과 같이 `cognitoedentity` SERVICE_ID를 예로 들 수 있습니다. 그러나 다음 테이블과 같이 일부 SERVICE_ID는 SDK 간에 다릅니다.

v1 열 중 하나에 있는 boldface SERVICE_ID는 v2에 사용된 SERVICE_ID와 다르다는 것을 나타냅니다.

서비스 이름	v1 패키지 이름	v1 artifactId	v2 artifactId	v2 패키지 이름
	모든 패키지 이름은 첫 번째 행에 표시된 대로 `com.amazonaws.services` 로 시작합니다.	모든 artifactId는 첫 번째 행과 같이 태그로 묶입니다.	모든 artifactId는 첫 번째 행과 같이 태그로 묶입니다.	모든 패키지 이름은 첫 번째 행에 표시된 대로 `software.amazon.awssdk` 로 시작합니다.
API Gateway	com.amazonaws.services.apigateway	<artifactId>aws-java-sdk-api-gateway</artifactId>	<artifactId>apigateway</artifactId>	software.amazon.awssdk.services.apigateway
앱 레지스트리	appregistry	appregistry	servicecatalogappregistry	servicecatalogappregistry
Application Discovery	applicationdiscovery	discovery	applicationdiscovery	applicationdiscovery
증강형 AI 런타임	augmentedairuntime	augmentedairuntime	sagemakera2iruntime	sagemakera2iruntime
Certificate Manager	certificatemanager	acm	acm	acm
CloudControl API	cloudcontrolapi	cloudcontrolapi	cloudcontrol	cloudcontrol

서비스 이름	v1 패키지 이름	v1 artifactId	v2 artifactId	v2 패키지 이름
cloudsearch	cloudsearchv2	cloudsearch	cloudsearch	cloudsearch
CloudSearch 도메인	cloudsearchdomain	cloudsearch	cloudsearchdomain	cloudsearchdomain
CloudWatch Events	cloudwatchevents	이벤트	cloudwatchevents	cloudwatchevents
CloudWatch Evidently	cloudwatchevidently	cloudwatchevidently	evidently	evidently
CloudWatch Logs	로그	로그	cloudwatchlogs	cloudwatchlogs
CloudWatch 지표	지표	cloudwatchmetrics	cloudwatch	cloudwatch
CloudWatch Rum	cloudwatchrum	cloudwatchrum	rum	rum
Cognito 자격 증명 공급자	cognitoidp	cognitoidp	cognitoidentityprovider	cognitoidentityprovider
캠페인 연결	connectcampaign	connectcampaign	connectcampaigns	connectcampaigns
Wisdom 연결	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

서비스 이름	v1 패키지 이름	v1 artifactId	v2 artifactId	v2 패키지 이름
Elastic Map Reduce	elasticmapreduce	emr	emr	emr
Glue DataBrew	gluedatabrew	gluedatabrew	databrew	databrew
IAM Roles Anywhere	iamrolesanywhere	iamrolesanywhere	rolesanywhere	rolesanywhere
자격 증명 관리	identitymanagement	iam	iam	iam
IoT 데이터	iotdata	iot	iotdataplane	iotdataplane
Kinesis Analytics	kinesisanalytics	kinesis –	kinesisanalytics	kinesisanalytics
Kinesis Firehose	kinesisfirehose	kinesis –	firehose	firehose
Kinesis 비디오 신호 채널	kinesisvideosignalingchannels	kinesisvideosignalingchannels	kinesisvideosignaling	kinesisvideosignaling
Lex	lexruntime	lex	lexruntime	lexruntime
Lookout For Vision	lookoutforvision	lookoutforvision	lookoutvision	lookoutvision
Mainframe Modernization	mainframemodernization	mainframemodernization	m2	m2
Marketplace Metering	marketplacemetering	marketplacemeteringservice	marketplacemetering	marketplacemetering
관리형 Grafana	managedgrafana	managedgrafana	grafana	grafana
Mechanical Turk	mturk	mechanicalturkrequester	mturk	mturk

서비스 이름	v1 패키지 이름	v1 artifactId	v2 artifactId	v2 패키지 이름
Migration Hub 전략 권장 사항	migrationhubstrategyrecommendations	migrationhubstrategyrecommendations	migrationhubstrategy	migrationhubstrategy
Nimble 스튜디오	nimblestudio	nimblestudio	nimble	nimble
Private 5G	private5g	private5g	privatenetworks	privatenetworks
Prometheus	prometheus	prometheus	amp	amp
휴지통	recyclebin	recyclebin	rbin	rbin
Redshift 데이터 API	redshiftdataapi	redshiftdataapi	redshiftdata	redshiftdata
Route 53	route53domains	route53	route53domains	route53domains
Sage Maker Edge Manager	sagemakeredgemanager	sagemakeredgemanager	sagemakeredge	sagemakeredge
보안 토큰	securitytoken	sts	sts	sts
서버 마이그레이션	servermigration	servermigration	sms	sms
간단한 이메일	simpleemail	ses	ses	ses
간단한 이메일 V2	simpleemailv2	sesv2	sesv2	sesv2
Simple Systems Management	simplesystemsmanagement	ssm	ssm	ssm
단순 워크플로	simpleworkflow	simpleworkflow	swf	swf
Step Functions	stepfunctions	stepfunctions	sfn	sfn

AWS SDK for Java 1.x와 2.x의 차이점은 무엇인가요?

이 단원에서는 AWS SDK for Java 버전 1.x를 사용하는 애플리케이션을 버전 2.x로 변환할 때 알아야 할 주요 변경 사항에 대해 설명합니다.

패키지 이름 변경 사항

SDK for Java 1.x에서 SDK for Java 2.x로의 눈에 띄는 변화는 패키지 이름 변경입니다. 패키지 이름은 SDK 2.x에서 `software.amazon.awssdk`로 시작하는 반면 SDK 1.x에서는 `com.amazonaws`를 사용합니다.

이러한 동일한 이름이 Maven 아티팩트를 SDK 1.x에서 SDK 2.x로 구분합니다. SDK 2.x용 Maven 아티팩트는 `software.amazon.awssdk` groupId를 사용하는 반면 SDK 1.x는 `com.amazonaws` groupId를 사용합니다.

SDK 2.x 아티팩트만 사용하는 프로젝트에 대한 `com.amazonaws` 종속성이 코드에 필요한 경우가 몇 번 있습니다. 이에 대한 한 가지 예는 서버 측에서 작업하는 경우입니다. AWS Lambda 이 내용은 이 가이드 앞부분의 Apache Maven 프로젝트 설정 단원에 나와 있습니다.

> **Note**
>
> SDK 1.x의 여러 패키지 이름에는 다음이 포함됩니다. `v2` 이 `v2` 경우에 를 사용한다는 것은 일반적으로 패키지의 코드가 서비스 버전 2에서 작동하도록 타겟팅되었음을 의미합니다.
> 전체 패키지 이름이 로 `com.amazonaws` 시작되므로 SDK 1.x 구성 요소입니다. SDK 1.x에서 이러한 패키지 이름의 예는 다음과 같습니다.
>
> - `com.amazonaws.services.dynamodbv2`
> - `com.amazonaws.retry.v2`
> - `com.amazonaws.services.apigatewayv2`
> - `com.amazonaws.services.simpleemailv2`

프로젝트에 버전 2.x 추가

Maven이 AWS SDK for Java 2.x를 사용할 때 종속성을 관리에 권장되는 방법입니다. 프로젝트에 버전 2.x 구성 요소를 추가하려면 SDK에 대한 종속 항목으로 `pom.xml` 파일을 업데이트합니다.

Example

```
<dependencyManagement>
    <dependencies>
        <dependency>
          <groupId>software.amazon.awssdk</groupId>
          <artifactId>bom</artifactId>
          <version>2.27.21</version>
          <type>pom</type>
          <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>

<dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>dynamodb</artifactId>
    </dependency>
</dependencies>
```

또한 프로젝트를 버전 2.x로 마이그레이션할 때 버전 1.x 및 2.x 단계를 사용할 수 있습니다.

변경이 불가능한 POJO

클라이언트와 작업 요청 및 응답 객체는 이제 변경할 수 없으며 생성 후에는 변경할 수 없습니다. 요청 또는 응답 변수를 재사용하려면 새 객체를 만들어 해당 변수에 할당해야 합니다.

Example 1.x의 요청 객체 업데이트

```
DescribeAlarmsRequest request = new DescribeAlarmsRequest();
DescribeAlarmsResult response = cw.describeAlarms(request);

request.setNextToken(response.getNextToken());
```

Example 2.x의 요청 객체 업데이트

```
DescribeAlarmsRequest request = DescribeAlarmsRequest.builder().build();
DescribeAlarmsResponse response = cw.describeAlarms(request);

request = DescribeAlarmsRequest.builder()
```

```
        .nextToken(response.nextToken())
        .build();
```

setter 및 getter 메서드

AWS SDK for Java 2.x에서는 설정자 메서드 이름에 `set` 또는 `with` 접두사가 포함되지 않습니다. 예를 들어 `*.withEndpoint()`은 `*.endpoint()`입니다.

getter 메서드 이름은 `get` 접두사를 사용하지 않습니다.

Example 1.x에서 setter 메서드 사용

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
          .withRegion("us-east-1")
          .build();
```

Example 2.x에서 setter 메서드 사용

```
DynamoDbClient client = DynamoDbClient.builder()
          .region(Region.US_EAST_1)
          .build();
```

Example 1.x에서 getter 메서드 사용

```
String token = request.getNextToken();
```

Example 2.x에서 getter 메서드 사용

```
String token = request.nextToken();
```

모델 클래스 이름

서비스 응답을 나타내는 모델 클래스 이름은 v1이 사용하는 `Result` 대신 v2의 `Response`로 끝납니다.

Example v1에서 응답을 나타내는 클래스 이름

```
CreateApiKeyResult
```

```
AllocateAddressResult
```

Example v2에서 응답을 나타내는 클래스 이름

```
CreateApiKeyResponse
AllocateAddressResponse
```

라이브러리 및 유틸리티의 마이그레이션 상태

Java용 SDK 라이브러리 및 유틸리티

다음 표에는 SDK for Java에 대한 라이브러리 및 유틸리티의 마이그레이션 상태가 나열되어 있습니다.

버전 1.12.x 이름	버전 2.x 이름	버전 2.x 이상
DynamoDBMapper	DynamoDbEnhancedClient	2.12.0
Waiters	Waiters	2.15.0
CloudFrontUrlSigner와 CloudFrontCookieSigner	CloudFrontUtilities	2.18.33
TransferManager	S3TransferManager	2.19.0
EC2 Metadata 클라이언트	EC2 Metadata 클라이언트	2.19.29
S3 URI 파서	S3 URI 파서	2.20.41
IAM 정책 빌더	IAM 정책 빌더	2.20.126
S3 이벤트 알림	S3 이벤트 알림	2.25.11
Amazon SQS 클라이언트 측 버퍼링	Amazon SQS용 자동 요청 배치 처리 API	2.28.0
진행 상황 리스너	진행 상황 리스너	아직 릴리스되지 않음

관련 라이브러리

다음 표에는 별도로 출시되었지만 Java용 SDK 2.x와 호환되는 라이브러리가 나열되어 있습니다.

Java용 SDK 버전 2.x와 함께 사용되는 이름	버전 이후
Amazon S3 암호화 클라이언트	3.0.0[1]
AWS Database Encryption SDK for DynamoDB	3.0.0[2]

[1]Amazon S3 암호화 클라이언트

Amazon S3용 암호화 클라이언트는 다음 Maven 종속성을 사용하여 사용할 수 있습니다.

```
<dependency>
    <groupId>software.amazon.encryption.s3</groupId>
    <artifactId>amazon-s3-encryption-client-java</artifactId>
    <version>3.x</version>
</dependency>
```

[2]AWS Database Encryption SDK for DynamoDB

AWS Database Encryption SDK for DynamoDB는 다음 Maven 종속성을 사용하여 V2에서 사용할 수 있습니다.

```
<dependency>
    <groupId>software.amazon.cryptography</groupId>
    <artifactId>aws-database-encryption-sdk-dynamodb</artifactId>
    <version>3.x</version>
</dependency>
```

Java SDK의 v1에서 작동하는 DynamoDB용 암호화 라이브러리에 대한 정보는 AWS Database Encryption SDK 개발자 안내서(Java용 Amazon DynamoDB Encryption Client) 및 GitHub에서 확인할 수 있습니다.

Java SDK의 V2와 호환되는 DynamoDB 암호화 라이브러리에 대한 자세한 내용은 AWS Database Encryption SDK 개발자 안내서 및 GitHub 소스를 참조하세요.

암호화 라이브러리에 대한 마이그레이션 정보는 AWS Database Encryption SDK 개발자 안내서에서 확인할 수 있습니다.

라이브러리 및 유틸리티에 대한 마이그레이션 세부 정보

- Transfer Manager

- EC2 메타데이터 유틸리티
- CloudFront 사전 서명
- S3 URI 구문 분석
- DynamoDB 매핑/문서 API
- IAM 정책 빌더
- S3 이벤트 알림
- SDK 지표 게시(1.x 설명서, 2.x 설명서)

클라이언트 변경

클라이언트 빌더

클라이언트 빌더 메서드를 사용하여 모든 클라이언트를 생성해야 합니다. 생성자를 더 이상 사용할 수 없습니다.

Example버전 1.x에서 클라이언트 생성

```
AmazonDynamoDB ddbClient = AmazonDynamoDBClientBuilder.defaultClient();
AmazonDynamoDBClient ddbClient = new AmazonDynamoDBClient();
```

Example버전 2.x에서 클라이언트 생성

```
DynamoDbClient ddbClient = DynamoDbClient.create();
DynamoDbClient ddbClient = DynamoDbClient.builder().build();
```

클라이언트 클래스 이름

이제 모든 클라이언트 클래스 이름은 완전히 카멜 대소문자로 표기되며 더 이상 접두사가 붙지 않습니다. Amazon 이러한 변경 내용은 에서 사용된 이름과 일치합니다. AWS CLI

Example 1.x의 클래스 이름

```
AmazonDynamoDB
AWSACMPCAAsyncClient
```

Example 2.x의 클래스 이름

```
DynamoDbClient
```

```
AcmAsyncClient
```

리전 클래스 이름 변경 사항

1.x 클라이언트	2.x 클라이언트
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
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

1.x 클라이언트	2.x 클라이언트
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
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

1.x 클라이언트	2.x 클라이언트
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
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

1.x 클라이언트	2.x 클라이언트
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
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

1.x 클라이언트	2.x 클라이언트
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
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

1.x 클라이언트	2.x 클라이언트
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
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

1.x 클라이언트	2.x 클라이언트
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
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

1.x 클라이언트	2.x 클라이언트
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
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

1.x 클라이언트	2.x 클라이언트
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
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

1.x 클라이언트	2.x 클라이언트
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
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

1.x 클라이언트	2.x 클라이언트
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
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

1.x 클라이언트	2.x 클라이언트
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
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

1.x 클라이언트	2.x 클라이언트
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
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

1.x 클라이언트	2.x 클라이언트
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
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	– ## ##
com.amazonaws.services.importexport.AmazonImportExportClient	– ## ##
com.amazonaws.services.inspector.AmazonInspectorAsyncClient	software.amazon.awssdk.services.inspector.InspectorAsyncClient
com.amazonaws.services.inspector.AmazonInspectorClient	software.amazon.awssdk.services.inspector.InspectorClient

1.x 클라이언트	2.x 클라이언트
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
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

1.x 클라이언트	2.x 클라이언트
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
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

1.x 클라이언트	2.x 클라이언트
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
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	지원되지 않음
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

1.x 클라이언트	2.x 클라이언트
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
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

1.x 클라이언트	2.x 클라이언트
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
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

1.x 클라이언트	2.x 클라이언트
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
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

1.x 클라이언트	2.x 클라이언트
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
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

1.x 클라이언트	2.x 클라이언트
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
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

1.x 클라이언트	2.x 클라이언트
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
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

1.x 클라이언트	2.x 클라이언트
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
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

1.x 클라이언트	2.x 클라이언트
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
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

1.x 클라이언트	2.x 클라이언트
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
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

1.x 클라이언트	2.x 클라이언트
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
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

1.x 클라이언트	2.x 클라이언트
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
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

1.x 클라이언트	2.x 클라이언트
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
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

1.x 클라이언트	2.x 클라이언트
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
com.amazonaws.services.xray.AWSXRayAsyncClient	software.amazon.awssdk.services.xray.XRayAsyncClient
com.amazonaws.services.xray.AWSXRayClient	software.amazon.awssdk.services.xray.XRayClient

클라이언트 만들기 기본값

버전 2.x에서는 기본 클라이언트 만들기 로직이 다음과 같이 변경되었습니다.

- S3의 기본 자격 증명 공급자 체인에는 더 이상 익명 자격 증명이 포함되지 않습니다. AnonymousCredentialsProvider를 사용하여 수동으로 S3에 대한 익명 액세스를 지정해야 합니다.
- 기본 클라이언트 만들기와 관련된 다음 환경 변수가 상이합니다.

1.x	2.x
AWS_CBOR_DISABLED	CBOR_ENABLED
AWS_ION_BINARY_DISABLE	BINARY_ION_ENABLED

- 기본 클라이언트 만들기와 관련된 다음 환경 시스템 속성이 상이합니다.

1.x	2.x
com.amazonaws.sdk.disableEc2Metadata	aws.disableEc2Metadata
com.amazonaws.sdk.ec2MetadataServiceEndpointOverride	aws.ec2MetadataServiceEndpoint
com.amazonaws.sdk.disableCbor	aws.cborEnabled
com.amazonaws.sdk.disableIonBinary	aws.binaryIonEnabled

- 버전 2.x는 다음 시스템 속성을 지원하지 않습니다.

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

- 사용자 지정 endpoints.json 파일에서 리전 구성을 로드하는 것은 더 이상 지원되지 않습니다.

클라이언트 구성

1.x에서는 클라이언트 또는 클라이언트 빌더에서 ClientConfiguration 인스턴스를 설정하여 SDK 클라이언트 구성을 수정했습니다. 버전 2.x에서는 클라이언트 구성이 별도의 구성 클래스로 분할됩니다. 별도의 구성 클래스를 사용하면 비동기 클라이언트와 동기 클라이언트에 대해

서로 다른 HTTP 클라이언트를 구성할 수 있지만 여전히 동일한 클래스를 사용할 수 있습니다. `ClientOverrideConfiguration`

Example버전 1.x의 클라이언트 구성

```
AmazonDynamoDBClientBuilder.standard()
.withClientConfiguration(clientConfiguration)
.build()
```

Example버전 2.x의 동기식 클라이언트 구성

```
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();
```

Example버전 2.x의 동기식 클라이언트 구성

```
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 클라이언트

주목할 만한 변경 사항

- 버전 2.x에서는 `clientBuilder.httpClientBuilder`를 사용하여 구현을 지정하여 런타임에 사용할 HTTP 클라이언트를 변경할 수 있습니다.
- `clientBuilder.httpClient`를 사용하여 HTTP 클라이언트를 서비스 클라이언트 빌더에 전달하면 서비스 클라이언트가 닫힐 때 기본적으로 HTTP 클라이언트는 닫히지 않습니다. 이렇게 하면 서비스 클라이언트 간에 HTTP 클라이언트를 공유할 수 있습니다.
- 비동기식 HTTP 클라이언트는 이제 비차단 I/O를 사용합니다.
- 이제 일부 작업에서는 성능 향상을 위해 HTTP/2를 사용합니다.

설정 변경

설정	1.x	2.x 동기식, Apache	2.x 비동기식, Netty
	`ClientConfiguration clientConfig = new ClientConfiguration()`	`ApacheHttpClient.Builder httpClientBuilder = ApacheHttpClient.builder()`	`NettyNioAsyncHttpClient.Builder httpClientBuilder = NettyNioAsyncHttpClient.builder()`
최대 연결 수	`clientConfig.setMaxConnections(...)` `clientConfig.withMaxConnections(...)`	`httpClientBuilder.maxConnections(...)`	`httpClientBuilder.maxConcurrency(...)`
연결 제한 시간	`clientConfig.setCo`	`httpClientBuilder.`	`httpClientBuilder.`

설정	1.x	2.x 동기식, Apache	2.x 비동기식, Netty
	`nnectionT imeout(...) clientConfig.wi thConnect ionTimeout(...)`	`connectio nTimeout(...) httpClientBui lder.conn ectionAcq uisitionT imeout(...)`	`connectio nTimeout(...)`
소켓 제한 시간	`clientCon fig.setSo cketTimeo ut(...) clientConf ig.withSo cketTimeo ut(...)`	`httpClien tBuilder. socketTim eout(...)`	`httpClien tBuilder. writeTime out(...) httpClien tBuilder. readTimeo ut(...)`
연결 시 TTL	`clientCon fig.setCo nnectionT TL(...) clientConf ig.withCo nnectionT TL(...)`	`httpClien tBuilder. connectio nTimeToLi ve(...)`	`httpClien tBuilder. connectio nTimeToLi ve(...)`
연결 시 최대 유휴	`clientCon fig.setCo nnectionM axIdleMil lis(...) clientCon fig.withC onnection MaxIdleMi llis(...)`	`httpClien tBuilder. connectio nMaxIdleT ime(...)`	`httpClien tBuilder. connectio nMaxIdleT ime(...)`

설정	1.x	2.x 동기식, Apache	2.x 비동기식, Netty
비활성화 후 유효성 확인	`clientConfig.setValidateAfterInactivityMillis(...)` `clientConfig.withValidateAfterInactivityMillis(...)`	비지원(요청 기능)	비지원(요청 기능)
로컬 주소	`clientConfig.setLocalAddress(...)` `clientConfig.withLocalAddress(...)`	`httpClientBuilder.localAddress(...)`	지원되지 않음
Expect-continue 사용	`clientConfig.setUseExpectContinue(...)` `clientConfig.withUseExpectContinue(...)`	`httpClientBuilder.expectContinueEnabled(...)`	비지원(요청 기능)
연결 리퍼	`clientConfig.setUseReaper(...)` `clientConfig.withReaper(...)`	`httpClientBuilder.useIdleConnectionReaper(...)`	`httpClientBuilder.useIdleConnectionReaper(...)`

설정	1.x	2.x 동기식, Apache	2.x 비동기식, Netty
	`AmazonDynamoDBClientBuilder.standard() .withClientConfiguration(clientConfiguration) .build()`	`DynamoDbClient.builder() .httpClientBuilder(httpClientBuilder) .build()`	`DynamoDbAsyncClient.builder() .httpClientBuilder(httpClientBuilder) .build()`

HTTP 클라이언트 프록시

설정	1.x	2.x 동기식, Apache	2.x 비동기식, Netty
	`ClientConfiguration clientConfig = new ClientConfiguration()`	`ProxyConfiguration.Builder proxyConfig = ProxyConfiguration.builder()`	`ProxyConfiguration.Builder proxyConfig = ProxyConfiguration.builder()`
프록시 호스트	`clientConfig.setProxyHost(...) clientConfig.withProxyHost(...)`	`proxyConfig.endpoint(...)`	`proxyConfig.host(...)`
프록시 포트	`clientConfig.setProxyPort(...) clientConfig.withProxyPort(...)`	`proxyConfig.endpoint(...)`	`proxyConfig.port(...)`

설정	1.x	2.x 동기식, Apache	2.x 비동기식, Netty
		프록시 포트는 endpoint에 포함되어 있습니다.	
프록시 사용자 이름	`clientConfig.setProxyUsername(...)` `clientConfig.withProxyUsername(...)`	`proxyConfig.username(...)`	`proxyConfig.username(...)`
프록시 비밀번호	`clientConfig.setProxyPassword(...)` `clientConfig.withProxyPassword(...)`	`proxyConfig.password(...)`	`proxyConfig.password(...)`
프록시 도메인	`clientConfig.setProxyDomain(...)` `clientConfig.withProxyDomain(...)`	`proxyConfig.ntlmDomain(...)`	비지원(요청 기능)
프록시 워크스테이션	`clientConfig.setProxyWorkspace(...)` `clientConfig.withProxyWorkstation(...)`	`proxyConfig.ntlmWorkstation(...)`	비지원(요청 기능)

설정	1.x	2.x 동기식, Apache	2.x 비동기식, Netty
프록시 인증 메서드	`clientConfig.setProxyAuthenticationMethods(...)` `clientConfig.withProxyAuthenticationMethods(...)`	지원되지 않음	비지원(요청 기능)
선제적 기본 프록시 인증	`clientConfig.setPreemptiveBasicProxyAuth(...)` `clientConfig.withPreemptiveBasicProxyAuth(...)`	`proxyConfig.preemptiveBasicAuthenticationEnabled(...)`	비지원(요청 기능)
비프록시 호스트	`clientConfig.setNonProxyHosts(...)` `clientConfig.withNonProxyHosts(...)`	`proxyConfig.nonProxyHosts(...)`	`proxyConfig.nonProxyHosts(...)`

설정	1.x	2.x 동기식, Apache	2.x 비동기식, Netty
소켓 프록시 사용 해제	`clientConfig.setDisableSocketProxy(...)` `clientConfig.withDisableSocketProxy(...)`	비지원(요청 기능)	비지원(요청 기능)

	`AmazonDynamoDBClientBuilder.standard()`
	`.withClientConfiguration(clientConfiguration)`
	`.build()`
		`httpClientBuilder.proxyConfiguration(`
		`proxyConfig.build())`
			`httpClientBuilder.proxyConfiguration(`
			`proxyConfig.build())`

클라이언트 재정의

설정	1.x	2.x
	`ClientConfiguration clientConfig =` `new ClientConfiguration()`	`ClientOverrideConfiguration.Builder overrideConfig =` `ClientOverrideConfiguration.builder()`
사용자 에이전트 접두사	`clientConfig.setUserAgentPrefix(...)` `clientConfig.withUserAgentPrefix(...)`	`overrideConfig.advancedOption(` `SdkAdvancedClientOption.USER_AGENT_PREFIX, ...)`

설정	1.x	2.x
사용자 에이전트 접미사	`clientConfig.setUserAgentSuffix(...)` `clientConfig.withUserAgentSuffix(...)`	`overrideConfig.advancedOption( SdkAdvancedClientOption.USER_AGENT_SUFFIX, ...)`
Signer	`clientConfig.setSignerOverride(...)` `clientConfig.withSignerOverride(...)`	`overrideConfig.advancedOption( SdkAdvancedClientOption.SIGNER, ...)`
추가 헤더	`clientConfig.addHeader(...)` `clientConfig.withHeader(...)`	`overrideConfig.putHeader(...)`
요청 제한 시간	`clientConfig.setRequestTimeout(...)` `clientConfig.withRequestTimeout(...)`	`overrideConfig.apiCallAttemptTimeout(...)`
클라이언트 실행 제한 시간	`clientConfig.setClientExecutionTimeout(...)` `clientConfig.withClientExecutionTimeout(...)`	`overrideConfig.apiCallTimeout(...)`
Gzip 사용	`clientConfig.setUseGzip(...)` `clientConfig.withGzip(...)`	비지원(요청 기능)

설정	1.x	2.x
소켓 버퍼 크기 힌트	`clientConfig.setSocketBufferSizeHints(...)` `clientConfig.withSocketBufferSizeHints(...)`	비지원(요청 기능)
캐시 응답 메타데이터	`clientConfig.setCacheResponseMetadata(...)` `clientConfig.withCacheResponseMetadata(...)`	비지원(요청 기능)
응답 메타데이터 캐시 크기	`clientConfig.setResponseMetadataCacheSize(...)` `clientConfig.withResponseMetadataCacheSize(...)`	비지원(요청 기능)
DNS 해석기	`clientConfig.setDnsResolver(...)` `clientConfig.withDnsResolver(...)`	비지원(요청 기능)

TCP 킵얼라이브	`clientConfig.setUseTcpKeepAlive(...)`
	`clientConfig.withTcpKeepAlive(...)`
		이제 이 옵션은 HTTP 클라이언트 구성에 포함됩니다.
		`- ApacheHttpClient.builder().tcpKeepAlive(true)`
		`- NettyNioAsyncHttpClient.builder().tcpKeepAlive(true)`

설정	1.x	2.x
보안 임의	`clientConfig.setSecureRandom(...)` `clientConfig.withSecureRandom(...)`	비지원(요청 기능)
	`AmazonDynamoDBClientBuilder.standard()` `.withClientConfiguration(clientConfiguration)` `.build()`	`DynamoDbClient.builder()` `.overrideConfiguration(overrideConfig.build())` `.build()`

클라이언트 재정의 재시도

설정	1.x	2.x
	`ClientConfiguration clientConfig = new ClientConfiguration()`	`ClientOverrideConfiguration.Builder overrideConfigBuilder = ClientOverrideConfiguration.builder();`
최대 오류 재시도	`clientConfig.setMaxErrorRetry(...)` `clientConfig.withMaxErrorRetry(...)`	`// Configure the default retry strategy.` `overrideConfigBuilder.retryStrategy(b -> b.maxAttempts(...));`
스로틀링 재시도 사용	`clientConfig.setUseThrottleRetries(...)`	지원되지 않음

설정	1.x	2.x
	`clientConfig.withUseThrottleRetries(...)`	
스로틀링 전 최대 연속 재시도 횟수	`clientConfig.setMaxConsecutiveRetriesBeforeThrottling(...)` `clientConfig.withMaxConsecutiveRetriesBeforeThrottling(...)`	지원되지 않음

	`AmazonDynamoDBClientBuilder.standard()`
	`.withClientConfiguration(clientConfiguration)`
	`.build()`
		`DynamoDbClient.builder()`
		`.overrideConfiguration(overrideConfigBuilder.build())`
		`.build();`
		
		`// You also have the option to use a lambda expression to configure and`
		`// build the 'ClientOverrideConfiguration.Builder'.`
		`DynamoDbClient client = DynamoDbClient.builder()`
		`.overrideConfiguration(o -> o.retryStrategy(b -> b.maxAttempts(5)))`
		`.build();`

비동기 클라이언트

설정	1.x	2.x
		`ClientAsyncConfiguration.Builder asyncConfig = ClientAsyncConfiguration.builder()`
실행자	`AmazonDynamoDBAsyncClientBuilder.standard() .withExecutorFactory(...) .build()`	`asyncConfig.advancedOption( SdkAdvancedAsyncClientOption.FUTURE_COMPLETION_EXECUTOR, ...)`
		`DynamoDbAsyncClient.builder() .asyncConfiguration(asyncConfig) .build()`

기타 클라이언트 변경 사항

1.x의 다음 `ClientConfiguration` 옵션은 SDK의 2.x에서 변경되었으며 동등한 직접 기능이 없습니다.

설정	1.x	2.x 동등
프로토콜	`clientConfig.setProtocol(Protocol.HTTP) clientConfig.withProtocol(Protocol.HTTP)`	프로토콜 설정은 기본적으로 HTTPS입니다. 설정을 수정하려면 클라이언트 빌더에서 HTTP 엔드포인트를 설정하는 프로토콜을 지정합니다.

설정	1.x	2.x 동등
		`clientBuilder.endpointOverride(URI.create("http://..."))`

자격 증명 공급자 변경 사항

이 단원에서는 AWS SDK for Java의 버전 1.x와 2.x 사이의 자격 증명 공급자 클래스 및 메서드의 이름 변경 사항을 매핑합니다.

주요 차이점

- 기본 자격 증명 공급자는 버전 2.x의 환경 변수보다 먼저 시스템 속성을 로드합니다. 자세한 정보는 자격 증명 사용하기를 참조하세요.
- 생성자 메서드는 `create` 또는 `builder` 메서드로 대체됩니다.

  Example

  ```
  DefaultCredentialsProvider.create();
  ```

- 비동기 새로 고침은 더 이상 기본적으로 설정되지 않습니다. 자격 증명 공급자의 `builder`를 사용해 지정해야 합니다.

  Example

  ```
  ContainerCredentialsProvider provider = ContainerCredentialsProvider.builder()
            .asyncCredentialUpdateEnabled(true)
            .build();
  ```

- `ProfileCredentialsProvider.builder()`를 사용하여 사용자 지정 프로필 파일의 경로를 지정할 수 있습니다.

  Example

  ```
  ProfileCredentialsProvider profile = ProfileCredentialsProvider.builder()

   .profileFile(ProfileFile.builder().content(Paths.get("myProfileFile.file")).build())
  ```

```
            .build();
```

- 프로필 파일 형식이 AWS CLI과 더 비슷하도록 변경되었습니다. 지침을 보려면 AWS Command Line Interface 사용 설명서의 AWS CLI 구성을 참조하세요.

버전 1.x와 2.x 간에 매핑된 자격 증명 공급자 변경 사항

`AWSCredentialsProvider`

변경 범주	1.x	2.x
패키지/클래스 이름	`com.amazonaws.auth.AWSCredentialsProvider`	`software.amazon.awssdk.auth.credentials.AwsCredentialsProvider`
메서드 이름	`getCredentials`	`resolveCredentials`
지원되지 않는 메서드	`refresh`	지원되지 않음

`DefaultAWSCredentialsProviderChain`

변경 범주	1.x	2.x
패키지/클래스 이름	`com.amazonaws.auth.DefaultAWSCredentialsProviderChain`	`software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider`
생성	`new DefaultAWSCredentialsProviderChain`	`DefaultCredentialsProvider.create`
지원되지 않는 메서드	`getInstance`	지원되지 않음
외부 설정의 우선순위	시스템 속성보다 환경 변수 우선	환경 변수보다 시스템 속성 우선

AWSStaticCredentialsProvider

변경 범주	1.x	2.x
패키지/클래스 이름	`com.amazonaws.auth.AWSStaticCredentialsProvider`	`software.amazon.awssdk.auth.credentials.StaticCredentialsProvider`
생성	`new AWSStaticCredentialsProvider`	`StaticCredentialsProvider.create`

EnvironmentVariableCredentialsProvider

변경 범주	1.x	2.x
패키지/클래스 이름	`com.amazonaws.auth.EnvironmentVariableCredentialsProvider`	`software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider`
생성	`new EnvironmentVariableCredentialsProvider`	`EnvironmentVariableCredentialsProvider.create`
환경 변수 이름	`AWS_ACCESS_KEY`	`AWS_ACCESS_KEY_ID`
	`AWS_SECRET_KEY`	`AWS_SECRET_ACCESS_KEY`

SystemPropertiesCredentialsProvider

변경 범주	1.x	2.x
패키지/클래스 이름	`com.amazonaws.auth.SystemPropertiesCredentialsProvider`	`software.amazon.awssdk.auth.credentials.SystemPropertyCredentialsProvider`
생성	`new SystemPropertiesCredentialsProvider`	`SystemPropertiesCredentialsProvider.create`
시스템 속성 이름	`aws.secretKey`	`aws.secretAccessKey`

ProfileCredentialsProvider

변경 범주	1.x	2.x
패키지/클래스 이름	`com.amazonaws.auth.profile.ProfileCredentialsProvider`	`software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider`
생성	`new ProfileCredentialsProvider`	`ProfileCredentialsProvider.create`
사용자 지정 프로필의 위치	• `AWS_CREDENTIAL_PROFILES_FILE` 환경 변수 • `new ProfileCredentialsProvider`	• `AWS_SHARED_CREDENTIALS_FILE` 환경 변수 • `ProfileCredentialsProvider.builder`

ContainerCredentialsProvider

변경 범주	1.x	2.x
패키지/클래스 이름	`com.amazonaws.auth.ContainerCredentialsProvider`	`software.amazon.awssdk.auth.credentials.ContainerCredentialsProvider`
생성	`new ContainerCredentialsProvider`	`ContainerCredentialsProvider.create`
비동기식 새로 고침 지정	지원되지 않음	기본 동작

InstanceProfileCredentialsProvider

변경 범주	1.x	2.x
패키지/클래스 이름	`com.amazonaws.auth.InstanceProfileCredentialsProvider`	`software.amazon.awssdk.auth.credentials.InstanceProfileCredentialsProvider`
생성	`new InstanceProfileCredentialsProvider`	`InstanceProfileCredentialsProvider.create`
비동기식 새로 고침 지정	`new InstanceProfileCredentialsProvider(true)`	`InstanceProfileCredentialProvider.builder().asyncCredentialUpdateEnabled(true).build()`
시스템 속성 이름	`com.amazonaws.sdk.disableEc2Metadata`	`aws.disableEc2Metadata`

변경 범주	1.x	2.x
	`com.amazonaws.sdk.ec2MetadataServiceEndpointOverride`	`aws.ec2MetadataServiceEndpoint`

`STSAssumeRoleSessionCredentialsProvider`

변경 범주	1.x	2.x
패키지/클래스 이름	`com.amazonaws.auth.STSAssumeRoleSessionCredentialsProvider`	`software.amazon.awssdk.services.sts.auth.StsAssumeRoleCredentialsProvider`
생성	• `new STSAssumeRoleSessionCredentialsProvider` • `new STSAssumeRoleSessionCredentialsProvider.Builder`	`StsAssumeRoleCredentialsProvider.builder`
비동기식 새로 고침	기본 동작	기본 동작
구성	`new STSAssumeRoleSessionCredentialsProvider.Builder`	`StsClient` 및 `AssumeRoleRequest` 요청 구성

STSSessionCredentialsProvider

변경 범주	1.x	2.x
패키지/클래스 이름	`com.amazonaws.auth.STSSessionCredentialsProvider`	`software.amazon.awssdk.services.sts.auth.StsGetSessionTokenCredentialsProvider`
생성	`new STSSessionCredentialsProvider`	`StsGetSessionTokenCredentialsProvider.builder`
비동기식 새로 고침	기본 동작	`StsGetSessionTokenCredentialsProvider.builder`
구성	생성자 파라미터	빌더에서 `StsClient` 및 `GetSessionTokenRequest` 요청 구성

WebIdentityFederationSessionCredentialsProvider

변경 범주	1.x	2.x
패키지/클래스 이름	`com.amazonaws.auth.WebIdentityFederationSessionCredentialsProvider`	`software.amazon.awssdk.services.sts.auth.StsAssumeRoleWithWebIdentityCredentialsProvider`
생성	`new WebIdentityFederationSessionCredentialsProvider`	`StsAssumeRoleWithWebIdentityCredentialsProvider.builder`

변경 범주	1.x	2.x
비동기식 새로 고침	기본 동작	`StsAssumeRoleWithWebIdentityCredentialsProvider.builder`
구성	생성자 파라미터	빌더에서 `StsClient` 및 `AssumeRoleWithWebIdentityRequest` 요청 구성

클래스 교체됨

1.x 클래스	2.x 대체 클래스
`com.amazonaws.auth.EC2ContainerCredentialsProviderWrapper`	`software.amazon.awssdk.auth.credentials.ContainerCredentialsProvider` 및 `software.amazon.awssdk.auth.credentials.InstanceProfileCredentialsProvider`
`com.amazonaws.services.s3.S3CredentialsProviderChain`	`software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider` 및 `software.amazon.awssdk.auth.credentials.AnonymousCredentialsProvider`

클래스 제거됨

1.x 클래스
`com.amazonaws.auth.ClasspathPropertiesFileCredentialsProvider`
`com.amazonaws.auth.PropertiesFileCredentialsProvider`

리전 변경

이 단원에서는 Region 및 Regions 클래스 사용을 위해 AWS SDK for Java 2.x에 구현된 변경 사항에 대해 설명합니다.

리전 구성

- 일부 AWS 서비스에는 리전별 엔드포인트가 없습니다. 이러한 서비스를 사용할 때는 리전을 Region.AWS_GLOBAL 또는 Region.AWS_CN_GLOBAL로 설정해야 합니다.

  Example

  ```
  Region region = Region.AWS_GLOBAL;
  ```

- com.amazonaws.regions.Regions 및 com.amazonaws.regions.Region 클래스가 이제 하나의 클래스 software.amazon.awssdk.regions.Region로 결합되었습니다.

메서드 및 클래스 이름 매핑

다음 표는 AWS SDK for Java의 버전 1.x와 2.x 사이의 리전 관련 클래스를 매핑합니다. of() 메서드를 사용하여 이러한 클래스의 인스턴스를 만들 수 있습니다.

Example

```
RegionMetadata regionMetadata = RegionMetadata.of(Region.US_EAST_1);
```

1.x 리전 클래스 메서드 변경 사항

1.x	2.x
Regions.fromName	Region.of
Regions.getName	Region.id
Regions.getDescription	Region.metadata().description()
Regions.getCurrentRegion	지원되지 않음
Regions.DEFAULT_REGION	지원되지 않음
Regions.name	Region.id

1.x 리전 클래스 메서드 변경 사항

1.x	2.x
`Region.getName`	`Region.id`
`Region.hasHttpsEndpoint`	지원되지 않음
`Region.hasHttpEndpoint`	지원되지 않음
`Region.getAvailableEndpoints`	지원되지 않음
`Region.createClient`	지원되지 않음

RegionMetadata 클래스 메서드 변경 사항

1.x	2.x
`RegionMetadata.getName`	`RegionMetadata.name`
`RegionMetadata.getDomain`	`RegionMetadata.domain`
`RegionMetadata.getPartition`	`RegionMetadata.partition`

ServiceMetadata 클래스 메서드 변경 사항

1.x	2.x
`Region.getServiceEndpoint`	`ServiceMetadata.endpointFor(Region)`
`Region.isServiceSupported`	`ServiceMetadata.regions().contains(Region)`

작업, 요청 및 응답 변경 사항

SDK for Java 버전 2에서는 요청이 클라이언트 작업으로 전달됩니다. 예를 들어 `DynamoDbClient's PutItemRequest`는 `DynamoDbClient.putItem` 작업으로 전달됩니다. 이 작업은 AWS 서비스에서 `PutItemResponse` 등의 응답을 반환합니다.

SDK for Java 버전 2는 버전 1에서 다음과 같이 변경되었습니다.

- 응답 페이지가 여러 개인 작업에는 이제 응답의 모든 항목을 자동으로 반복할 수 있는 `Paginator` 메서드가 포함됩니다.
- 요청 및 응답은 변경할 수 없습니다.
- 생성자 대신 정적 빌더 메서드를 사용하여 요청 및 응답을 만들어야 합니다. 예를 들어 버전 1의 `new PutItemRequest().withTableName(...)`은 이제 `PutItemRequest.builder().tableName(...).build()`입니다.
- 작업은 요청(`dynamoDbClient.putItem(request -> request.tableName(...))`)을 만드는 간단한 방법을 지원합니다.

다음 섹션에서는 버전 1과 버전 2 간의 특정 변경 사항을 설명합니다. 일부 파라미터 유형 변경 사항은 마이그레이션 도구를 사용하여 자동으로 변환할 수 있지만, 다른 변경 사항은 코드를 수동으로 업데이트해야 합니다.

데이터 파라미터 변경

버전 1에서는 많은 작업에서 시간 기반 파라미터에 대한 `java.util.Date` 객체를 수락했습니다. 버전 2에서는 이러한 작업에서 대신 `java.time.Instant` 객체를 사용합니다.

마이그레이션 도구를 사용하여 `Date` 파라미터를 자동으로 변환하거나 `Date` 객체에서 `toInstant()` 메서드를 호출하여 파라미터를 수동으로 변환할 수 있습니다.

Example - 버전 1에서 만료 날짜가 있는 미리 서명된 URL 생성

```
// Generate a presigned URL that expires at a specific date
Date expiration = new Date(System.currentTimeMillis() + 3600000); // 1 hour from now
URL presignedUrl = s3Client.generatePresignedUrl(bucketName, keyName, expiration);
```

Example - 버전 2에서 만료 인스턴스가 있는 미리 서명된 URL 생성

```
// Generate a presigned URL that expires at a specific instant
Date expiration = new Date(System.currentTimeMillis() + 3600000); // 1 hour from now
PresignedGetObjectRequest presignedRequest = presigner.presignGetObject(
    GetObjectPresignRequest.builder()
        .getObjectRequest(GetObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
```

```
            .build())
        .signatureDuration(Duration.between(Instant.now(), expiration.toInstant()))
        .build());
```

바이너리 데이터 처리 변경

버전 1에서는 `ByteBuffer` 객체를 직접 사용하여 바이너리 데이터를 처리했습니다. 버전 2에서 SDK는 바이너리 데이터로 작업하는 보다 편리하고 유형에 안전한 방법을 제공하는 `SdkBytes` 객체를 사용합니다.

마이그레이션 도구를 사용하여 `SdkBytes`를 자동으로 `ByteBuffer`로 변환하거나 반환된 `SdkBytes` 객체에서 `asByteBuffer()`를 호출하여 수동으로 변환할 수 있습니다.

Example - 버전 1의 메시지 속성에서 바이너리 데이터 가져오기

```
// Get binary data from a message attribute
MessageAttributeValue messageAttributeValue = new MessageAttributeValue();
ByteBuffer binaryValue = messageAttributeValue.getBinaryValue();
String binaryString = new String(messageAttributeValue.getBinaryValue().array());
```

Example - 버전 2의 메시지 속성에서 바이너리 데이터 가져오기

```
// Get binary data from a message attribute
MessageAttributeValue messageAttributeValue = MessageAttributeValue.builder().build();
ByteBuffer binaryValue = messageAttributeValue.binaryValue().asByteBuffer();
String binaryString = new
 String(messageAttributeValue.binaryValue().asByteBuffer().array());
```

제한 시간 파라미터 변경 사항

버전 1에서는 제한 시간 값이 밀리초를 나타내는 정수 값으로 지정되었습니다. 버전 2에서 제한 시간 파라미터는 `java.time.Duration` 객체를 사용하여 유형 안전과 명확성을 개선합니다.

마이그레이션 도구를 사용하여 숫자 제한 시간 값을 자동으로 변환하거나 숫자 값을 적절한 `Duration` 팩토리 메서드로 래핑하여 수동으로 변환할 수 있습니다.

Example - 버전 1에서 요청 제한 시간 설정

```
// Set request timeout in milliseconds
ClientConfiguration clientConfiguration = new ClientConfiguration();
```

```
clientConfiguration.setRequestTimeout(5000); // 5 seconds
```

Example - 버전 2에서 요청 제한 시간 설정

```
// Set request timeout using Duration
ClientConfiguration clientConfiguration = new ClientConfiguration();
clientConfiguration.setRequestTimeout(Duration.ofMillis(5000)); // 5 seconds

// Or more clearly:
clientConfiguration.setRequestTimeout(Duration.ofSeconds(5)); // 5 seconds
```

제한 시간 값에 다음 `Duration` 팩토리 메서드를 사용할 수 있습니다.

- `Duration.ofMillis(long millis)` - 밀리초 값의 경우.
- `Duration.ofSeconds(long seconds)` - 두 번째 값의 경우.
- `Duration.ofMinutes(long minutes)` - 분 값의 경우.

AWS SDK for Java의 1.x와 2.x 간의 스트리밍 작업 차이점

Amazon S3 `getObject` 및 `putObject` 메서드 등의 스트리밍 작업은 SDK 버전 2.x에서 비차단 I/O를 지원합니다. 따라서 요청 및 응답 모델 객체는 더 이상 `InputStream`을 파라미터로 사용하지 않습니다. 대신 동기식 요청의 경우 요청 객체가 `RequestBody`(바이트 스트림)를 허용합니다. 동등한 비동기식 요청은 `AsyncRequestBody`를 허용합니다.

Example 1.x에서의 아마존 S3 **`putObject`** 작업

```
s3client.putObject(BUCKET, KEY, new File(file_path));
```

Example 2.x에서의 Amazon S3 **`putObject`** 작업

```
s3client.putObject(PutObjectRequest.builder()
                                   .bucket(BUCKET)
                                   .key(KEY)
                                   .build(),
                   RequestBody.of(Paths.get("myfile.in")));
```

스트리밍 응답 객체는 V2에서 동기식 클라이언트의 경우 `ResponseTransformer`, 비동기식 클라이언트의 경우 `AsyncResponseTransformer`를 허용합니다.

Example 1.x에서의 아마존 S3 **getObject** 작업

```
S3Object o = s3.getObject(bucket, key);
S3ObjectInputStream s3is = o.getObjectContent();
FileOutputStream fos = new FileOutputStream(new File(key));
```

Example 2.x에서의 Amazon S3 **getObject** 작업

```
s3client.getObject(GetObjectRequest.builder().bucket(bucket).key(key).build(),
  ResponseTransformer.toFile(Paths.get("key")));
```

SDK for Java 2.x에서 스트리밍 응답 작업에는 응답을 메모리에 로드하고 인 메모리의 일반적인 유형 변환을 간소화하는 `AsBytes` 메서드가 포함됩니다.

AWS SDK for Java의 1.x와 2.x 간 직렬화 차이점

파라미터 차이를 요청할 객체 나열

SDK for Java v1와 v2.x는 파라미터를 요청하기 위해 List 객체를 직렬화하는 방식이 다릅니다.

SDK for Java 1.x는 빈 목록을 직렬화하지 않는 반면, SDK for Java 2.x는 빈 목록을 빈 파라미터로 직렬화합니다.

예를 들어 `SampleRequest`를 받는 `SampleOperation`이 있는 서비스를 생각해 보겠습니다. `SampleRequest`는 다음 예시와 같이 두 개의 파라미터(문자열 유형 `str1`과 목록 유형 `listParam`)를 허용합니다.

Example 1.x의 **SampleOperation**

```
SampleRequest v1Request = new SampleRequest()
    .withStr1("TestName");

sampleServiceV1Client.sampleOperation(v1Request);
```

와이어 레벨 로깅은 `listParam` 파라미터가 직렬화되지 않았음을 보여줍니다.

```
Action=SampleOperation&Version=2011-01-01&str1=TestName
```

Example 2.x의 **SampleOperation**

```
sampleServiceV2Client.sampleOperation(b -> b
```

```
    .str1("TestName"));
```

와이어 레벨 로깅은 `listParam` 파라미터가 값 없이 직렬화되었음을 보여줍니다.

```
Action=SampleOperation&Version=2011-01-01&str1=TestName&listParam=
```

V1의 POJO와 V2의 빌더 비교

V1 SDK for Java는 변경 가능한 POJO 클래스를 사용하기 때문에 Jackson과 같은 직렬화 및 역직렬화 라이브러리는 모델 객체를 직접 사용할 수 있습니다.

반대로 V2 SDK for Java는 변경 불가능한 모델 객체를 사용합니다. 중간 빌더를 사용하여 역직렬화를 수행해야 합니다.

다음 예제에서는 Jackson `ObjectMapper`를 사용하여 V1 및 V2로 `headBucket` API 호출을 직렬화/역직렬화할 때의 차이점을 보여줍니다.

```
    public void sendRequest() throws IOException {
        final String bucketName = "amzn-s3-demo-bucket";
        final ObjectMapper mapper = new ObjectMapper();

        // V1 uses POJOs to serialize and deserialize.
        final AmazonS3 v1S3Client = AmazonS3ClientBuilder.defaultClient();
        HeadBucketResult resultV1 = v1S3Client.headBucket(
            new HeadBucketRequest(bucketName));

        String v1Serialized = mapper.writeValueAsString(resultV1);

        HeadBucketResult deserializedV1 = mapper.readValue(v1Serialized,
HeadBucketResult.class);

        // V2 uses builders to serialize and deserialize.
        S3Client v2S3Client = S3Client.create();
        HeadBucketResponse v2Response = v2S3Client.headBucket(
            b -> b.bucket(bucketName));

        String v2Serialized = mapper.writeValueAsString(
            v2Response.toBuilder());

        HeadBucketResponse v2Deserialized = mapper.readValue(
            v2Serialized, HeadBucketResponse.serializableBuilderClass())
            .build();
```

```
    }
```

AWS SDK for Java의 1.x와 2.x 간 역직렬화 차이점

V2의 비어 있는 컬렉션과 V1의 **nulls** 비교

SDK for Java v1.x 및 v2.x는 JSON 응답을 비어 있는 목록 및 맵과 역직렬화하는 방식이 다릅니다.

SDK가 목록 또는 맵으로 모델링된 누락 속성이 있는 응답을 수신하면 V1은 누락된 속성을 null로 역직렬화하는 반면, V2는 속성을 변경할 수 없는 비어 있는 컬렉션 객체로 역직렬화합니다.

예를 들어 DynamoDB 클라이언트에서 `describeTable` 메서드에 대해 반환된 응답을 고려합니다. 다음 예제 메서드에는 글로벌 보조 인덱스가 없는 테이블에서 `describeTable` 메서드를 실행하는 V2 DynamoDB 클라이언트와 V1 DynamoDB 클라이언트가 포함되어 있습니다.

Example응답에 누락된 목록으로 모델링된 속성의 역직렬화

```
public void deserializationDiffs(){

    DescribeTableResponse v2Response = dynamoDbClientV2.describeTable(builder ->
 builder.tableName(TABLE_NAME));
    // V2 provides has* methods on model objects for list/map members. No null check
 needed.
    LOGGER.info( String.valueOf(v2Response.table().hasGlobalSecondaryIndexes()) );

 LOGGER.info( String.valueOf(v2Response.table().globalSecondaryIndexes().isEmpty()) );
    // V2 deserialize to an empty collection.
    LOGGER.info(v2Response.table().globalSecondaryIndexes().toString());

    // V1 deserialize to null.
    DescribeTableResult v1Result = dynamoDbClientV1.describeTable(new
 DescribeTableRequest(TABLE_NAME));
    if (v1Result.getTable().getGlobalSecondaryIndexes() != null){
        LOGGER.info(v1Result.getTable().getGlobalSecondaryIndexes().toString());
    } else {
        LOGGER.info("The list of global secondary indexes returned by the V1 call is
 <null>");
    }
}
```

다음은 로그된 출력을 보여줍니다.

```
INFO  org.example.DeserializationDifferences:45 - false
```

```
INFO  org.example.DeserializationDifferences:46 - true
INFO  org.example.DeserializationDifferences:48 - []
INFO  org.example.DeserializationDifferences:55 - The list of global secondary indexes
 returned by the V1 call is <null>
```

Java SDK 2.x는 null을 반환하는 대신 비어 있는 목록을 역직렬화하고 변경 불가능한 비어 있는 컬렉션에 매핑하여 더 안전하고 간결한 코드를 승격해 관용적인 접근 방식을 취합니다. V2를 사용하면 서비스가 이전 예제에 표시된 hasGlobalSecondaryIndexes와 같이 has* 메서드를 사용하여 목록 또는 맵으로 모델링된 속성을 반환했는지 확인할 수 있습니다.

이 접근 방식은 명시적 null 확인의 필요성을 방지하고 존재하지 않거나 비어 있는 데이터 구문을 처리하기 위한 최신 Java 모범 사례에 부합합니다.

전체 예제

```
package org.example;

import com.amazonaws.regions.Regions;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.model.DescribeTableRequest;
import com.amazonaws.services.dynamodbv2.model.DescribeTableResult;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.BillingMode;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;

import java.util.UUID;

public class DeserializationDifferences {
    private static final Logger LOGGER =
 LoggerFactory.getLogger(DeserializationDifferences.class);
    private static final String TABLE_NAME = "DeserializationTable-" +
 UUID.randomUUID();
    DynamoDbClient dynamoDbClientV2 = DynamoDbClient.create();
    AmazonDynamoDB dynamoDbClientV1 =
 AmazonDynamoDBClientBuilder.standard().withRegion(Regions.US_EAST_1).build();

    public static void main(String[] args) {
```

```
        DeserializationDifferences difference = new DeserializationDifferences();
        difference.createTable();
        difference.deserializationDiffs();
        difference.deleteTable();
    }

    public void createTable(){
        dynamoDbClientV2.createTable(b -> b
                .billingMode(BillingMode.PAY_PER_REQUEST)
                .tableName(TABLE_NAME)
                .keySchema(b1 -> b1.attributeName("Id").keyType(KeyType.HASH))
                .attributeDefinitions(b2 ->
b2.attributeName("Id").attributeType(ScalarAttributeType.S)));
        dynamoDbClientV2.waiter().waitUntilTableExists(b -> b.tableName(TABLE_NAME));
    }

    public void deserializationDiffs(){

        DescribeTableResponse v2Response = dynamoDbClientV2.describeTable(builder ->
builder.tableName(TABLE_NAME));
        // V2 provides has* methods on model objects for list/map members. No null
check needed.
        LOGGER.info( String.valueOf(v2Response.table().hasGlobalSecondaryIndexes()) );

LOGGER.info( String.valueOf(v2Response.table().globalSecondaryIndexes().isEmpty()) );
        // V2 deserialize to an empty collection.
        LOGGER.info(v2Response.table().globalSecondaryIndexes().toString());

        // V1 deserialize to null.
        DescribeTableResult v1Result = dynamoDbClientV1.describeTable(new
DescribeTableRequest(TABLE_NAME));
        if (v1Result.getTable().getGlobalSecondaryIndexes() != null){
            LOGGER.info(v1Result.getTable().getGlobalSecondaryIndexes().toString());
        } else {
            LOGGER.info("The list of global secondary indexes returned by the V1 call
is <null>");
        }
    }

    public void deleteTable(){
        dynamoDbClientV2.deleteTable(b -> b.tableName(TABLE_NAME));
        dynamoDbClientV2.waiter().waitUntilTableNotExists(b ->
b.tableName(TABLE_NAME));
    }
```

```
}
```

V1 및 V2 클라이언트의 `describeTable` 메서드에 대한 JSON 응답에는 `GlobalSecondaryIndexes` 속성이 없습니다.

```
{
  "Table": {
    "AttributeDefinitions": [
      {
        "AttributeName": "Id",
        "AttributeType": "S"
      }
    ],
    "BillingModeSummary": {
      "BillingMode": "PAY_PER_REQUEST",
      "LastUpdateToPayPerRequestDateTime": ...
    },
    "CreationDateTime": ...,
    "DeletionProtectionEnabled": false,
    "ItemCount": 0,
    "KeySchema": [
      {
        "AttributeName": "Id",
        "KeyType": "HASH"
      }
    ],
    "ProvisionedThroughput": {
      "NumberOfDecreasesToday": 0,
      "ReadCapacityUnits": 0,
      "WriteCapacityUnits": 0
    },
    "TableArn": "arn:aws:dynamodb:us-east-1:11111111111:table/
DeserializationTable-...",
    "TableId": "...",
    "TableName": "DeserializationTable-...",
    "TableSizeBytes": 0,
    "TableStatus": "ACTIVE",
    "TableThroughputModeSummary": {
      "LastUpdateToPayPerRequestDateTime": ...,
      "TableThroughputMode": "PAY_PER_REQUEST"
    },
    "WarmThroughput": {
      "ReadUnitsPerSecond": 12000,
```

```
      "Status": "ACTIVE",
      "WriteUnitsPerSecond": 4000
    }
  }
}
```

예외 변경

예외 클래스 이름, 구문 및 관계가 변경되었습니다.
`software.amazon.awssdk.core.exception.SdkException`은 다른 모든 예외가 확장되는 새 기본 `Exception` 클래스입니다.

이 표는 예외 클래스 이름 변경 내용을 매핑합니다.

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`

다음 표는 버전 1.x와 2.x 사이의 예외 클래스에 대한 메서드를 매핑합니다.

1.x	2.x
`AmazonServiceException.getRequestId`	`SdkServiceException.requestId`
`AmazonServiceException.getServiceName`	`AwsServiceException.awsErrorDetails().serviceName`

1.x	2.x
`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`

서비스별 CLI

Amazon S3 변경 사항

DK for Java 2.x는 기본적으로 익명 액세스를 사용 해제합니다. 따라서 `AnonymousCredentialsProvider`를 사용하여 익명 액세스를 사용해야 합니다.

작업 이름 변경 사항

AWS SDK for Java 2.x에서는 Amazon S3 클라이언트의 많은 작업 이름이 변경되었습니다. 버전 1.x에서는 서비스 API에서 Amazon S3 클라이언트가 직접 생성되지 않습니다. 이로 인해 SDK 작업과 서비스 API 간에 불일치가 발생합니다. 버전 2.x에서는 이제 서비스 API와의 일관성을 높이기 위해 Amazon S3 클라이언트를 생성합니다.

다음 테이블에는 두 버전의 작업 이름이 나와 있습니다.

Amazon S3 작업 이름

1.x	2.x
abortMultipartUpload	abortMultipartUpload
changeObjectStorageClass	copyObject
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

1.x	2.x
deleteObject	deleteObject
deleteObjectTagging	deleteObjectTagging
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 또는 getBucketLifecycleConfiguration
getBucketLocation	getBucketLocation
getBucketLoggingConfiguration	getBucketLogging

1.x	2.x
getBucketMetricsConfiguration	getBucketMetricsConfiguration
getBucketNotificationConfiguration	getBucketNotification 또는 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

1.x	2.x
listBucketAnalyticsConfigurations	listBucketAnalyticsConfigurations
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

1.x	2.x
setBucketCrossOriginConfiguration	putBucketCors
setBucketEncryption	putBucketEncryption
setBucketInventoryConfiguration	putBucketInventoryConfiguration
setBucketLifecycleConfiguration	putBucketLifecycle 또는 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 변경 사항

SNS 클라이언트는 더 이상 액세스하도록 구성된 리전 이외의 리전에서 SNS 주제에 액세스할 수 없습니다.

Amazon SQS 변경 사항

SQS 클라이언트는 더 이상 액세스하도록 구성된 리전 이외의 리전에서 SQS 대기열에 액세스할 수 없습니다.

Amazon RDS 변경 사항

SDK for Java 2.x는 1.x의 `RdsIamAuthTokenGenerator` 클래스 대신 `RdsUtilities#generateAuthenticationToken`을 사용합니다.

AWS SDK for Java의 버전 1에서 버전 2로 Amazon S3 작업 시 변경 사항

AWS SDK for Java 2.x는 새 패키지 구문, 업데이트된 클래스 이름, 수정된 메서드 서명을 포함하여 S3 클라이언트에 중요한 변경 사항을 도입합니다. 마이그레이션 도구를 사용하여 V1에서 V2로 많은 메서드를 자동으로 마이그레이션할 수 있지만, `listNextBatchOfObjects` 및 `selectObjectContent`와 같은 일부 메서드는 수동 마이그레이션이 필요합니다. 또한 V2는 `AccessControlList` 및 `CannedAccessControlList`와 같은 특정 V1 클래스를 새 구현으로 대체합니다.

주제

- AWS SDK for Java의 버전 1과 버전 2 간의 S3 클라이언트 차이점
- Transfer Manager를 AWS SDK for Java의 버전 1에서 버전 2로 마이그레이션
- 버전 1에서 버전 2로 Amazon S3 URI 구문 분석에 관한 변경 사항
- S3 이벤트 알림 API를 버전 1에서 버전 2로 변경

AWS SDK for Java의 버전 1과 버전 2 간의 S3 클라이언트 차이점

이 주제에서는 SDK for Java 버전 1과 버전 2의 S3 클라이언트 간 차이점이 마이그레이션 도구가 마이그레이션을 자동화하는 방법에 따라 구성됩니다. 이 도구는 대부분의 메서드를 V1에서 V2로 마이그레이션하는 것을 지원하지만 일부 메서드는 수동으로 마이그레이션해야 합니다. S3 클라이언트 메서드 외에도 일부 S3 V1 클래스에는 수동으로 마이그레이션해야 하는 직접 V2와 동등한 요소가 없습니다.

목차

마이그레이션 도구에서 지원하는 V1 메서드 예제

마이그레이션 도구는 대부분의 메서드를 V1에서 V2로 자동으로 마이그레이션합니다. `putObject` 및 `getObject` 메서드는 2가지 예제입니다.

putObject

```
// SDK V1
```

```
s3Client.putObject("amzn-s3-demo-bucket", "my-key", "Hello World!");


// SDK V2
s3Client.putObject(req -> req
    .bucket("amzn-s3-demo-bucket")
    .key("my-key"),
    RequestBody.fromString("Hello World!"));
```

getObject

```
// SDK V1
S3Object object = s3Client.getObject("amzn-s3-demo-bucket", "my-key");
InputStream content = object.getObjectContent();


// SDK V2
ResponseInputStream<GetObjectResponse> response = s3Client.getObject(req -> req
    .bucket("amzn-s3-demo-bucket")
    .key("my-key"));
```

수동 마이그레이션이 필요한 V1 메서드

다음 V1 S3 클라이언트 메서드를 수동으로 마이그레이션해야 합니다. 마이그레이션 도구를 사용하면 V2 출력 Java 파일에 이 주제로 연결되는 설명이 표시됩니다.

V1의 **S3ObjectId**에서 V2의 **GetObjectRequest.builder()**를 사용한 V1 **getObject**

```
// SDK V1
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard()
        .withRegion(Regions.US_WEST_2)
        .build();

S3ObjectId s3ObjectId = new S3ObjectId(
    "amzn-s3-demo-bucket",
    "object-key",
    "abc123version"
);

GetObjectRequest getRequest= new GetObjectRequest(s3ObjectId);
S3Object s3ObjectVersioned = s3Client.getObject(getRequest);
```

```
// SDK V2
// V2 does not include a 'S3ObjectId' class. V2 uses the request builder pattern
// to supply the bucket, key, and version parameters.
S3Client s3Client = S3Client.builder()
    .region(Region.US_WEST_2)
    .build();

GetObjectRequest getRequest = GetObjectRequest.builder()
    .bucket("amzn-s3-demo-bucket")
    .key("object-key")
    .versionId("abc123version")
    .build();

ResponseInputStream<GetObjectResponse> response = s3Client.getObject(getRequest);
```

V1의 **listNextBatchOfObjects**에서 V2의 **listObjectsV2Paginator**

```
// SDK V1
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .build();

ObjectListing objectListing = s3ClientV1.listObjects("bucket-name");
while (objectListing.isTruncated()) {
    objectListing = s3ClientV1.listNextBatchOfObjects(objectListing);
    for (S3ObjectSummary summary : objectListing.getObjectSummaries()) {
        System.out.println(summary.getKey());
    }
}


// SDK V2
S3Client s3ClientV2 = S3Client.builder()
    .region(Region.US_WEST_2)
    .build();

ListObjectsV2Request request = ListObjectsV2Request.builder()
    .bucket("bucket-name")
    .build();

// V2 returns a paginator.
ListObjectsV2Iterable responses = s3Client.listObjectsV2Paginator(request);
```

```
for (ListObjectsV2Response page : responses) {
    page.contents().forEach(content -> {
        System.out.println(content.key());
    });
}
```

V1의 **listNextBatchOfVersions**에서 V2의 **listObjectVersionsPaginator**

```
// SDK V1
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .build();

VersionListing versionListing = s3ClientV1.listVersions("bucket-name", "prefix");
while (versionListing.isTruncated()) {
    versionListing = s3ClientV1.listNextBatchOfVersions(versionListing);
    for (S3VersionSummary version : versionListing.getVersionSummaries()) {
        System.out.println(version.getKey() + " " + version.getVersionId());
    }
}


// SDK V2
S3Client s3ClientV2 = S3Client.builder()
    .region(Region.US_WEST_2)
    .build();

ListObjectVersionsRequest request = ListObjectVersionsRequest.builder()
    .bucket("bucket-name")
    .prefix("prefix")
    .build();

// V2 returns a paginator.
ListObjectVersionsIterable responses = s3ClientV2.listObjectVersionsPaginator(request);

for (ListObjectVersionsResponse page : responses) {
    page.versions().forEach(version -> {
        System.out.println(version.key() + " " + version.versionId());
    });
}
```

`selectObjectContent`

```
// SDK V1
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .build();

SelectObjectContentRequest request = new SelectObjectContentRequest()
    .withBucket("bucket-name")
    .withKey("object-key")
    .withExpression("select * from S3Object")
    .withExpressionType(ExpressionType.SQL)


SelectObjectContentResult result = s3ClientV1.selectObjectContent(request);
InputStream resultInputStream = result.getPayload().getRecordsInputStream();


// SDK V2
// In V2, 'selectObjectContent()' is available only on the S3AsyncClient.
// V2 handles responses using an event-based 'SelectObjectContentEventStream'.
S3AsyncClient s3ClientV2 = S3AsyncClient.builder()
    .region(Region.US_WEST_2)
    .build();

SelectObjectContentRequest request = SelectObjectContentRequest.builder()
    .bucket("bucket-name")
    .key("object-key")
    .expression("select * from S3Object")
    .expressionType(ExpressionType.SQL)
    .build();

SelectObjectContentResponseHandler handler = new SelectObjectContentResponseHandler() {
    // Implement the required abstract methods such as 'onEventStream()'.
};

CompletableFuture<Void> future = s3ClientV2.selectObjectContent(request, handler);
// The 'SelectObjectContentResponseHandler' implementation processes the results.
```

`PutBucketAclRequest.builder()`에 V1의 `setBucketAcl`에서 V2의 `acl` 메서드

```
// SDK V1
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard()
```

```
    .withRegion(Regions.US_WEST_2)
    .build();

AccessControlList acl = new AccessControlList();
acl.grantPermission(GroupGrantee.AllUsers, Permission.Read);
s3ClientV1.setBucketAcl("bucket-name", acl);


// SDK V2
S3Client s3ClientV2 = S3Client.builder()
    .region(Region.US_WEST_2)
    .build();

PutBucketAclRequest request = PutBucketAclRequest.builder()
    .bucket("bucket-name")
    .acl(BucketCannedACL.PRIVATE)
    .build();

s3ClientV2.putBucketAcl(request);
```

PutObjectAclRequest.builder()에 V1의 **setObjectAcl**에서 V2의 **acl** 메서드

```
// SDK V1
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .build();

AccessControlList acl = new AccessControlList();
acl.grantPermission(GroupGrantee.AllUsers, Permission.Read);
s3ClientV1.setObjectAcl("bucket-name", "object-key", acl);


// SDK V2
S3Client s3ClientV2 = S3Client.builder()
    .region(Region.US_WEST_2)
    .build();

PutObjectAclRequest request = PutObjectAclRequest.builder()
    .bucket("bucket-name")
    .key("object-key")
    .acl(ObjectCannedACL.PRIVATE)
    .build();
```

```
s3ClientV2.putObjectAcl(request);
```

V1의 **initiateMultipartUpload**에서 V2의 **createMultipartUpload**

마이그레이션 예제

```
// SDK V1
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .build();

ObjectMetadata metadata = new ObjectMetadata();
metadata.setContentType("application/zip");
metadata.addUserMetadata("mykey", "myvalue");

InitiateMultipartUploadRequest initRequest = new InitiateMultipartUploadRequest(
    "bucket-name",
    "object-key",
    metadata
);

InitiateMultipartUploadResult initResponse =
 s3ClientV1.initiateMultipartUpload(initRequest);
String uploadId = initResponse.getUploadId();


// SDK V2
// V1 uses ObjectMetadata methods, whereas V2 uses a simple Map.
S3Client s3ClientV2 = S3Client.builder()
    .region(Region.US_WEST_2)
    .build();

CreateMultipartUploadRequest createMultipartRequest =
 CreateMultipartUploadRequest.builder()
    .bucket("amzn-s3-demo-bucket")
    .key("object-key")
    .contentType("application/zip")
    .metadata(Collections.singletonMap("mykey", "myvalue"))
    .build();

CreateMultipartUploadResponse response =
 s3ClientV2.createMultipartUpload(createMultipartRequest);
String uploadId = response.uploadId();
```

구현 차이점

다음 메서드의 기본 Content-Type 헤더 값은 다음 테이블의 내용과 같이 다릅니다.

SDK 버전	메서드	기본값 **Content-Type**
버전 1	initiateMultipartUpload	application/octet-stream
버전 2	createMultipartUpload	binary/octet-stream

클라이언트 빌더에 V1의 **setRegion**에서 V2의 **region** 메서드

```
// SDK V1
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard().build();
s3ClientV1.setRegion(Region.getRegion(Regions.US_WEST_2));


// SDK V2
S3Client s3ClientV2 = S3Client.builder()
    .region(Region.US_WEST_2)
    .build();
```

V1의 **setS3ClientOptions(S3ClientOptions clientOptions)**

V2는 setS3ClientOptions 메서드와 함께 단일 S3ClientOptions 객체를 사용하는 대신 클라이언트 빌더에 옵션을 설정하는 메서드를 제공합니다.

V1의 **setBucketLoggingConfiguration**에서 V2의 **putBucketLogging**

```
// SDK V1
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .build();

BucketLoggingConfiguration loggingConfig = new BucketLoggingConfiguration();
loggingConfig.setDestinationBucketName("log-bucket");
```

```
SetBucketLoggingConfigurationRequest request = new
 SetBucketLoggingConfigurationRequest(
    "amzn-s3-demo-source-bucket",
    loggingConfig
);

s3ClientV1.setBucketLoggingConfiguration(request);


// SDK V2
// In V2, V1's 'BucketLoggingConfiguration' is replaced by 'BucketLoggingStatus'
// and 'LoggingEnabled'.
S3Client s3ClientV2 = S3Client.builder()
    .region(Region.US_WEST_2)
    .build();

LoggingEnabled loggingEnabled = LoggingEnabled.builder()
    .targetBucket("log-bucket")
    .build();

BucketLoggingStatus loggingStatus = BucketLoggingStatus.builder()
    .loggingEnabled(loggingEnabled)
    .build();

PutBucketLoggingRequest request = PutBucketLoggingRequest.builder()
    .bucket("amzn-s3-demo-source-bucket")
    .bucketLoggingStatus(loggingStatus)
    .build();

s3ClientV2.putBucketLogging(request);
```

V1의 **setBucketLifecycleConfiguration**에서 V2의 **putBucketLifecycleConfiguration**

```
// SDK V1
BucketLifecycleConfiguration.Rule rule = new BucketLifecycleConfiguration.Rule()
    .withId("Archive and Delete Rule")
    .withPrefix("documents/")
    .withStatus(BucketLifecycleConfiguration.ENABLED)
    .withTransitions(Arrays.asList(
        new Transition()
            .withDays(30)
            .withStorageClass(StorageClass.StandardInfrequentAccess.toString()),
        new Transition()
```

```
            .withDays(90)
            .withStorageClass(StorageClass.Glacier.toString())
    ))
    .withExpirationInDays(365);

BucketLifecycleConfiguration configuration = new BucketLifecycleConfiguration()
    .withRules(Arrays.asList(rule));

s3ClientV1.setBucketLifecycleConfiguration("amzn-s3-demo-bucket", configuration);


// SDK V2
LifecycleRule rule = LifecycleRule.builder()
    .id("Archive and Delete Rule")
    .filter(LifecycleRuleFilter.builder()
        .prefix("documents/")
        .build())
    .status(ExpirationStatus.ENABLED)
    .transitions(Arrays.asList(
        Transition.builder()
            .days(30)
            .storageClass(TransitionStorageClass.STANDARD_IA)
            .build(),
        Transition.builder()
            .days(90)
            .storageClass(TransitionStorageClass.GLACIER)
            .build()
    ))
    .expiration(LifecycleExpiration.builder()
        .days(365)
        .build())
    .build();

PutBucketLifecycleConfigurationRequest request =
 PutBucketLifecycleConfigurationRequest.builder()
    .bucket("amzn-s3-demo-bucket")
    .lifecycleConfiguration(BucketLifecycleConfiguration.builder()
        .rules(rule)
        .build())
    .build();

s3ClientV2.putBucketLifecycleConfiguration(request);
```

V1의 **setBucketTaggingConfiguration**에서 V2의 **putBucketTagging**

```
// SDK V1
List<TagSet> tagsets = new ArrayList<>();
TagSet tagSet = new TagSet();
tagSet.setTag("key1", "value1");
tagSet.setTag("key2", "value2");
tagsets.add(tagSet);

BucketTaggingConfiguration bucketTaggingConfiguration = new
 BucketTaggingConfiguration();
bucketTaggingConfiguration.setTagSets(tagsets);

SetBucketTaggingConfigurationRequest request = new
 SetBucketTaggingConfigurationRequest(
    "amzn-s3-demo-bucket",
    bucketTaggingConfiguration
);

s3ClientV1.setBucketTaggingConfiguration(request);


// SDK V2
Tagging tagging = Tagging.builder()
    .tagSet(Arrays.asList(
        Tag.builder()
            .key("key1")
            .value("value1")
            .build(),
        Tag.builder()
            .key("key2")
            .value("value2")
            .build()
    ))
    .build();

PutBucketTaggingRequest request = PutBucketTaggingRequest.builder()
    .bucket("amzn-s3-demo-bucket")
    .tagging(tagging)
    .build();

s3ClientV2.putBucketTagging(request);
```

수동 마이그레이션이 필요한 V1 클래스

V1의 **AccessControlList**에서 V2의 **AccessControlPolicy**

```
// SDK V1
AccessControlList aclV1 = new AccessControlList();
aclV1.setOwner(new Owner("owner-id", "owner-name"));
aclV1.grantPermission(GroupGrantee.AllUsers, Permission.Read);

// SDK V2
// To migrate from V1 to V2, replace direct 'AccessControlList' modifications with an
// 'AccessControlPolicy.builder()' that contains both owner information and grants.
// Note that V2's approach requires building the complete permission set upfront rather
 than modifying permissions incrementally.
AccessControlPolicy acpV2 = AccessControlPolicy.builder()
    .owner(Owner.builder()
        .id("owner-id")
        .displayName("owner-name")
        .build())
    .grants(Arrays.asList(
         Grant.builder()
            .grantee(Grantee.builder()
                 .type(Type.GROUP)
                 .uri("http://acs.amazonaws.com/groups/global/AllUsers")
                 .build())
             .permission(Permission.READ)
             .build()
     ))
     .build();
```

V1의 **CannedAccessControlList** 열거형에서 V2의 **BucketCannedACL** 및 **ObjectCannedACL** 열거형

```
// SDK V1
// In V1, 'CannedAccessControlList' is an enumeration of predefined ACLs.
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard().build();

// Creating a bucket.
s3ClientV1.setBucketAcl("bucket-name", CannedAccessControlList.PublicRead);

// Creating an object.
PutObjectRequest putObjectRequest = new PutObjectRequest("bucket-name", "object-key",
 file)
```

```
    .withCannedAcl(CannedAccessControlList.PublicRead);
s3ClientV1.putObject(putObjectRequest);


// SDK V2
// V2 replaces V1's 'CannedAccessControlList' with 'BucketCannedACL' for buckets and
 'ObjectCannedACL' for objects.
S3Client s3ClientV2 = S3Client.builder().build();

// Creating a bucket.
PutBucketAclRequest bucketRequest = PutBucketAclRequest.builder()
    .bucket("bucket-name")
    .acl(BucketCannedACL.PRIVATE)
    .build();
s3ClientV2.putBucketAcl(bucketRequest);

// Creating an object.
PutObjectRequest objectRequest = PutObjectRequest.builder()
    .bucket("bucket-name")
    .key("object-key")
    .acl(ObjectCannedACL.PUBLIC_READ)
    .build();
s3ClientV2.putObject(objectRequest, RequestBody.fromFile(file));
```

V1의 **BucketNotificationConfiguration**에서 V2의 **NotificationConfiguration**

```
//SDK V1
BucketNotificationConfiguration notificationConfig = new
 BucketNotificationConfiguration();

// Adding configurations by name
notificationConfig.addConfiguration("lambdaConfig",
    new LambdaConfiguration("arn:aws:lambda:function", "s3:ObjectCreated:"));

notificationConfig.addConfiguration("topicConfig",
    new TopicConfiguration("arn:aws:sns:topic", "s3:ObjectRemoved:"));

notificationConfig.addConfiguration("queueConfig",
    new QueueConfiguration("arn:aws:sqs:queue", "s3:ObjectRestore:*"));

s3Client.setBucketNotificationConfiguration("bucket", notificationConfig);
```

```
// SDK V2
// In V2, V1's BucketNotificationConfiguration is renamed to
 NotificationConfiguration.
// V2 contains no common abstract class for LambdaFunction/Topic/Queue configurations.
NotificationConfiguration notificationConfig = NotificationConfiguration.builder()
    .lambdaFunctionConfigurations(
        LambdaFunctionConfiguration.builder()
            .lambdaFunctionArn("arn:aws:lambda:function")
            .events(Event.valueOf("s3:ObjectCreated:"))
            .build())
    .topicConfigurations(
        TopicConfiguration.builder()
            .topicArn("arn:aws:sns:topic")
            .events(Event.valueOf("s3:ObjectRemoved:"))
            .build())
    .queueConfigurations(
        QueueConfiguration.builder()
            .queueArn("arn:aws:sqs:queue")
            .events(Event.valueOf("s3:ObjectRestore:*"))
            .build())
    .build();

s3Client.putBucketNotificationConfiguration(req -> req
    .bucket("bucket")
    .notificationConfiguration(notificationConfig));
```

요청 빌더에 V1의 **MultiFactorAuthentication**에서 V2의 **mfa** 메서드

```
// SDK V1
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .build();

BucketVersioningConfiguration versioningConfig = new BucketVersioningConfiguration()
    .withStatus(BucketVersioningConfiguration.ENABLED);

// Create an MFA configuration object.
MultiFactorAuthentication mfa = new MultiFactorAuthentication(
    "arn:aws:iam::1234567890:mfa/user",
    "123456"
);

// Create the request object.
```

```
SetBucketVersioningConfigurationRequest request = new
 SetBucketVersioningConfigurationRequest(
    "bucket-name",
    versioningConfig,
    mfa
);

// Send the request.
s3ClientV1.setBucketVersioningConfiguration(request);


// SDK V2
// V2 replaces V1's MultiFactorAuthentication POJO with parameters you set on the
 request builder.
S3Client s3ClientV2 = S3Client.builder()
    .region(Region.US_WEST_2)
    .build();

PutBucketVersioningRequest request = PutBucketVersioningRequest.builder()
    .bucket("bucket-name")
    .versioningConfiguration(VersioningConfiguration.builder()
        .status(BucketVersioningStatus.ENABLED)
        .build())
    .mfa("arn:aws:iam::1234567890:mfa/user 123456")  // Single method takes both MFA
 erial number and token.
    .build();

s3ClientV2.putBucketVersioning(request);
```

Transfer Manager를 AWS SDK for Java의 버전 1에서 버전 2로 마이그레이션

이 마이그레이션 안내서에서는 생성자 변경 사항, 메서드 매핑 및 일반적인 작업에 대한 코드 예제를 포함하여 Transfer Manager v1과 S3 Transfer Manager v2 간의 주요 차이점을 다룹니다. 이러한 차이점을 검토한 후 기존 Transfer Manager 코드를 성공적으로 마이그레이션하여 v2에서 향상된 성능과 비동기식 작업을 활용할 수 있습니다.

> ⚠ AWS SDK 마이그레이션 도구 정보
>
> AWS SDK for Java는 v1 Transfer Manager API의 대부분을 v2로 마이그레이션할 수 있는 자동화된 마이그레이션 도구를 제공합니다. 그러나 마이그레이션 도구는 여러 v1 Transfer Manager 기능을 지원하지 않습니다. 이러한 경우 이 주제의 안내를 사용하여 Transfer Manager 코드를 수동으로 마이그레이션해야 합니다.

이 안내서 전체에서 마이그레이션 상태는 마이그레이션 도구가 생성자, 메서드 또는 기능을 자동으로 마이그레이션할 수 있는지 여부를 보여줍니다.

- 지원됨: 마이그레이션 도구가 이 코드를 자동으로 변환할 수 있습니다.
- 지원되지 않음: 코드를 수동으로 마이그레이션해야 합니다.

'지원됨'으로 표시된 항목에 대해서도 마이그레이션 결과를 검토하고 철저히 테스트합니다. Transfer Manager 마이그레이션에는 동기식 작업에서 비동기식 작업으로의 상당한 아키텍처 변경이 포함됩니다.

개요

S3 Transfer Manager v2는 Transfer Manager API에 중요한 변경 사항을 도입합니다. S3 Transfer Manager v2는 비동기식 작업을 기반으로 하며 특히 AWS CRT 기반 Amazon S3 클라이언트를 사용할 때 더 나은 성능을 제공합니다.

주요 차이점

- 패키지: `com.amazonaws.services.s3.transfer` → `software.amazon.awssdk.transfer.s3`
- 클래스 이름: `TransferManager` → `S3TransferManager`
- 클라이언트 종속성: 동기식 Amazon S3 클라이언트 → 비동기식 Amazon S3 클라이언트 (`S3AsyncClient`)
- 아키텍처: 동기식 작업 → `CompletableFuture`를 사용한 비동기식 작업
- 성능: AWS CRT 기반 클라이언트 지원으로 향상

높은 수준의 변경 사항

속성	V1	V2
Maven 종속성	`aws-java-sdk-s3`	`s3-transfer-manager`
패키지:	`com.amazonaws.services.s3.transfer`	`software.amazon.awssdk.transfer.s3`
기본 클래스	`TransferManager`	`S3TransferManager`

속성	V1	V2
Amazon S3 클라이언트	AmazonS3(동기식)	S3AsyncClient (비동기식)
반환 유형	차단된 작업	CompletableFuture<T>

Maven 종속성

V1

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>com.amazonaws</
groupId>
            <artifactId>aws-java-sdk-
bom</artifactId>
            <version>> 1.12.787[1]</
version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
<dependencies>
    <dependency>
        <groupId>com.amazonaws</gro
upId>
        <artifactId>aws-java-sdk-s3
</artifactId>
    </dependency>
</dependencies>
```

V2

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>software.
amazon.awssdk</groupId>
            <artifactId>bom</a
rtifactId>
            <version> 2.31.68[2]</version
>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
<dependencies>
    <dependency>
        <groupId>software.amazon.aw
ssdk</groupId>
        <artifactId>s3-transfer-man
ager</artifactId>
    </dependency>
    <!-- Optional: For enhanced
 performance with AWS CRT -->
    <dependency>
        <groupId>software.amazon.aw
ssdk.crt</groupId>
        <artifactId>aws-crt</artifa
ctId>
        <version> 0.38.5[3]</version>
    </dependency>
```

V1	V2
	</dependencies>

[1] 최신 버전. [2] 최신 버전. [3]최신 버전.

클라이언트 생성자 마이그레이션

지원되는 생성자(자동 마이그레이션)

V1 생성자	V2와 동일	마이그레이션 상태
new TransferManager()	S3TransferManager.create()	지원
TransferManagerBuilder.defaultTransferManager()	S3TransferManager.create()	지원
TransferManagerBuilder.standard().build()	S3TransferManager.builder().build()	지원
new TransferManager(AWSCredentials)	S3TransferManager.builder() .s3Client(S3AsyncClient.builder() .credentialsProvider(...).build()) .build()	지원
new TransferManager(AWSCredentialsProvider)	S3TransferManager.builder() .s3Client(S3AsyncClient.builder() .credentialsProvider(...).build()) .build()	지원

지원되지 않는 생성자(수동 마이그레이션 필요)

V1 생성자	V2와 동일	마이그레이션 참고 사항
`new TransferManager(AmazonS3)`	수동 마이그레이션 필요	`S3AsyncClient` 별도로 만들기
`new TransferManager(AmazonS3, ExecutorService)`	수동 마이그레이션 필요	`S3AsyncClient` 만들기 및 실행기 구성
`new TransferManager(AmazonS3, ExecutorService, boolean)`	수동 마이그레이션 필요	지원되지 않는 `shutDownThreadPools` 파라미터

수동 마이그레이션 예제

V1 코드:

```
AmazonS3 s3Client = AmazonS3ClientBuilder.defaultClient();
TransferManager transferManager = new TransferManager(s3Client);
```

V2 코드:

```
// Create an `S3AsyncClient` with similar configuration
S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
    .credentialsProvider(DefaultCredentialsProvider.create())
    .build();

// Provide the configured `S3AsyncClient` to the S3 transfer manager builder.
S3TransferManager transferManager = S3TransferManager.builder()
    .s3Client(s3AsyncClient)
    .build();
```

클라이언트 메서드 마이그레이션

현재 마이그레이션 도구는 기본 `copy`, `download`, `upload`, `uploadDirectory`, `downloadDirectory`, `resumeDownload`, `resumeUpload` 메서드를 지원합니다.

주요 전송 메서드

V1 메서드	V2 메서드	반환 유형 변경 사항	마이그레이션 상태
`upload(String, String, File)`	`uploadFile(UploadFileRequest)`	`Upload` → `FileUpload`	지원
`upload(PutObjectRequest)`	`upload(UploadRequest)`	`Upload` → `Upload`	지원
`download(String, String, File)`	`downloadFile(DownloadFileRequest)`	`Download` → `FileDownload`	지원
`download(GetObjectRequest, File)`	`downloadFile(DownloadFileRequest)`	`Download` → `FileDownload`	지원
`copy(String, String, String, String)`	`copy(CopyRequest)`	`Copy` → `Copy`	지원
`copy(CopyObjectRequest)`	`copy(CopyRequest)`	`Copy` → `Copy`	지원
`uploadDirectory(String, String, File, boolean)`	`uploadDirectory( UploadDirectoryRequest)`	`MultipleFileUpload` → `DirectoryUpload`	지원
`downloadDirectory(String, String, File)`	`downloadDirectory( DownloadDirectoryRequest)`	`MultipleFileDownload` → `DirectoryDownload`	지원

재사용 가능한 전송 메서드

V1 메서드	V2 메서드	마이그레이션 상태
`resumeUpload(PersistableUpload)`	`resumeUploadFile(ResumableFileUpload)`	지원
`resumeDownload(PersistableDownload)`	`resumeDownloadFile(ResumableFileDownload)`	지원

수명 주기 메서드

V1 메서드	V2 메서드	마이그레이션 상태
`shutdownNow()`	`close()`	지원
`shutdownNow(boolean)`	`close()` 메서드를 사용하여 수동으로 코드 조정	지원되지 않음

지원되지 않는 V1 클라이언트 메서드

V1 메서드	V2 대안	Notes
`abortMultipartUploads(String, Date)`	하위 수준 Amazon S3 클라이언트 사용	지원되지 않음
`getAmazonS3Client()`	참조를 별도로 저장	지원되지 않음, v2에 getter 없음
`getConfiguration()`	참조를 별도로 저장	지원되지 않음, v2에 getter 없음
`uploadFileList(...)`	여러 `uploadFile()` 호출	지원되지 않음
`TransferStateChangeListener` 파라미터가 있는 copy 메서드	사용`TransferListener`	수동 마이그레이션 예제 참조

V1 메서드	V2 대안	Notes
`S3ProgressListener` 파라미터가 있는 download 메서드	사용`TransferListener`	수동 마이그레이션 예제 참조
파라미터가 4개 이상인 `downloadDirectory` 메서드		수동 마이그레이션 예제 참조
`ObjectMetadataProvider` 파라미터가 있는 upload 메서드	요청 시 메타데이터 설정	수동 마이그레이션 예제 참조
`*Provider` 파라미터가 있는 `uploadDirectory` 메서드	요청 시 태그 설정	수동 마이그레이션 예제 참조

`TransferStateChangeListener` 파라미터가 있는 `copy` 메서드

- `copy(CopyObjectRequest copyObjectRequest, AmazonS3 srcS3, TransferStateChangeListener stateChangeListener)`
- `copy(CopyObjectRequest copyObjectRequest, TransferStateChangeListener stateChangeListener)`

```
// V1
 ----------------------------------------------------------------------------------------------------
// Initialize source S3 client
AmazonS3 s3client = AmazonS3ClientBuilder.standard()
                .withRegion("us-west-2")
                .build();

// Initialize Transfer Manager
TransferManager tm = TransferManagerBuilder.standard()
                .withS3Client(srcS3)
                .build();

CopyObjectRequest copyObjectRequest = new CopyObjectRequest(
                "amzn-s3-demo-source-bucket",
                "source-key",
                "amzn-s3-demo-destination-bucket",
                "destination-key"
        );
```

```
TransferStateChangeListener stateChangeListener = new TransferStateChangeListener() {
            @Override
            public void transferStateChanged(Transfer transfer, TransferState state) {
              //Implementation of the TransferStateChangeListener
            }
        };

Copy copy = tm.copy(copyObjectRequest, srcS3, stateChangeListener);


// V2
 ---------------------------------------------------------------------------------------------------
S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
                .region(Region.US_WEST_2)
                .build();

S3TransferManager transferManager = S3TransferManager.builder()
                .s3Client(s3AsyncClient)
                .build();

// Create transfer listener (equivalent to TransferStateChangeListener in v1)

TransferListener transferListener = new TransferListener() {
            @Override
            public void transferInitiated(Context.TransferInitiated context) {
               //Implementation
               System.out.println("Transfer initiated");
            }

            @Override
            public void bytesTransferred(Context.BytesTransferred context) {
                //Implementation
                System.out.println("Bytes transferred");
            }

            @Override
            public void transferComplete(Context.TransferComplete context) {
                //Implementation
                System.out.println("Transfer completed!");
            }

            @Override
            public void transferFailed(Context.TransferFailed context) {
```

```
                //Implementation
                System.out.println("Transfer failed");
            }
        };

CopyRequest copyRequest = CopyRequest.builder()
                                .copyObjectRequest(req -> req
                                    .sourceBucket("amzn-s3-demo-source-bucket")
                                    .sourceKey("source-key")
                                    .destinationBucket("amzn-s3-demo-destination-bucket")
                                    .destinationKey("destination-key")
                                 )
                                  .addTransferListener(transferListener) // Configure the
 transferListener into the request
                                  .build();

Copy copy = transferManager.copy(copyRequest);
```

S3ProgressListener 파라미터가 있는 **download** 메서드

- download(GetObjectRequest getObjectRequest, File file, S3ProgressListener progressListener)
- download(GetObjectRequest getObjectRequest, File file, S3ProgressListener progressListener, long timeoutMillis)
- download(GetObjectRequest getObjectRequest, File file, S3ProgressListener progressListener, long timeoutMillis, boolean resumeOnRetry)

```
// V1
 ----------------------------------------------------------------------------------------
S3ProgressListener progressListener = new S3ProgressListener() {
        @Override
        public void progressChanged(com.amazonaws.event.ProgressEvent progressEvent) {
            long bytes = progressEvent.getBytesTransferred();
            ProgressEventType eventType = progressEvent.getEventType();
            // Use bytes and eventType as needed
        }

        @Override
        public void onPersistableTransfer(PersistableTransfer persistableTransfer) {

        }
```

```
    };

Download download1 = tm.download(getObjectRequest, file, progressListener);
Download download2 = tm.download(getObjectRequest, file, progressListener,
 timeoutMillis)
Download download3 = tm.download(getObjectRequest, file, progressListener,
 timeoutMillis, true)

// V2
 ---------------------------------------------------------------------------------------------
TransferListener transferListener = new TransferListener() {
    @Override
    public void transferInitiated(Context.InitializedContext context) {
        // Equivalent to ProgressEventType.TRANSFER_STARTED_EVENT
        System.out.println("Transfer initiated");
    }

    @Override
    public void bytesTransferred(Context.BytesTransferred context) {
        // Equivalent to ProgressEventType.REQUEST_BYTE_TRANSFER_EVENT
        long bytes = context.bytesTransferred();
        System.out.println("Bytes transferred: " + bytes);
    }

    @Override
    public void transferComplete(Context.TransferComplete context) {
        // Equivalent to ProgressEventType.TRANSFER_COMPLETED_EVENT
        System.out.println("Transfer completed");
    }

    @Override
    public void transferFailed(Context.TransferFailed context) {
        // Equivalent to ProgressEventType.TRANSFER_FAILED_EVENT
        System.out.println("Transfer failed: " + context.exception().getMessage());
    }
};
DownloadFileRequest downloadFileRequest =
                         DownloadFileRequest.builder()
                             .getObjectRequest(getObjectRequest)
                             .destination(file.toPath())
                             .addTransferListener(transferListener)
                             .build();

// For download1
```

```
FileDownload download = transferManager.downloadFile(downloadFileRequest);

// For download2
CompletedFileDownload completedFileDownload = download.completionFuture()
                                                      .get(timeoutMillis,
 TimeUnit.MILLISECONDS);

// For download3, the v2 SDK does not have a direct equiavalent to the `resumeOnRetry`
 method of v1.
// If a download is interrupted, you need to start a new download request.
```

파라미터가 4개 이상인 **downloadDirectory** 메서드

- downloadDirectory(String bucketName, String keyPrefix, File destinationDirectory, boolean resumeOnRetry)
- downloadDirectory(String bucketName, String keyPrefix, File destinationDirectory, boolean resumeOnRetry, KeyFilter filter)
- downloadDirectory(String bucketName, String keyPrefix, File destinationDirectory, KeyFilter filter)

```
// V1
 ----------------------------------------------------------------------------------------
KeyFilter filter = new KeyFilter() {
            @Override
            public boolean shouldInclude(S3ObjectSummary objectSummary) {
                //Filter implementation
            }
        };
MultipleFileDownload multipleFileDownload = tm.downloadDirectory(bucketName, keyPrefix,
 destinationDirectory, filter);

// V2
 ----------------------------------------------------------------------------------------
// The v2 SDK does not have a direct equiavalent to the `resumeOnRetry` method of v1.
// If a download is interrupted, you need to start a new download request.
DownloadFilter filter = new DownloadFilter() {
            @Override
            public boolean test(S3Object s3Object) {
                // Filter implementation.
            }
        };
```

```
DownloadDirectoryRequest downloadDirectoryRequest =
                                DownloadDirectoryRequest.builder()
                                    .bucket(bucketName)
                                    .filter(filter)
                                    .listObjectsV2RequestTransformer(builder ->
 builder.prefix(keyPrefix))
                                    .destination(destinationDirectory.toPath())
                                    .build();

DirectoryDownload directoryDownload =
 transferManager.downloadDirectory(downloadDirectoryRequest);
```

ObjectMetadata 파라미터가 있는 **upload** 메서드

- upload(String bucketName, String key, InputStream input, ObjectMetadata objectMetadata)

```
// V1
 ---------------------------------------------------------------------------------------------
ObjectMetadata metadata = new ObjectMetadata();
ObjectMetadata metadata = new ObjectMetadata();

metadata.setContentType("text/plain");        // System-defined metadata
metadata.setContentLength(22L);               // System-defined metadata
metadata.addUserMetadata("myKey", "myValue"); // User-defined metadata

PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName, key, inputStream,
 metadata);

Upload upload = transferManager.upload("amzn-s3-demo-bucket", "my-key", inputStream,
 metadata);

// V2
 ---------------------------------------------------------------------------------------------
/* When you use an InputStream to upload in V2, you should specify the content length
   and use `RequestBody.fromInputStream()`.
   If you don't provide the content length, the entire stream will be buffered in
 memory.
   If you can't determine the content length, we recommend using the CRT-based S3
 client.
*/
```

```
Map<String, String> userMetadata = new HashMap<>();
userMetadata.put("x-amz-meta-myKey", "myValue");

PutObjectRequest putObjectRequest =
                        PutObjectRequest.builder()
                            .bucket("amzn-s3-demo-bucket1")
                            .key("k")
                            .contentType("text/plain") //System-defined metadata
 usually has separate methods in the builder.
                            .contentLength(22L)
                            .metadata(userMetadata) //metadata() is only for user-
defined metadata.
                            .build();

UploadRequest uploadRequest =
                        UploadRequest.builder()
                            .putObjectRequest(putObjectRequest)
                            .requestBody(AsyncRequestBody.fromInputStream(stream, 22L,
 executor))
                            .build();

transferManager.upload(uploadRequest).completionFuture().join();
```

ObjectMetadataProvider 파라미터가 있는 **uploadDirectory**

- uploadDirectory(String bucketName, String virtualDirectoryKeyPrefix, File directory, boolean includeSubdirectories, ObjectMetadataProvider metadataProvider)
- uploadDirectory(String bucketName, String virtualDirectoryKeyPrefix, File directory, boolean includeSubdirectories, ObjectMetadataProvider metadataProvider, ObjectTaggingProvider taggingProvider)
- uploadDirectory(String bucketName, String virtualDirectoryKeyPrefix, File directory, boolean includeSubdirectories, ObjectMetadataProvider metadataProvider, ObjectTaggingProvider taggingProvider, ObjectCannedAclProvider cannedAclProvider)

```
// V1
 ----------------------------------------------------------------------------------------
tm.uploadDirectory(bucketName, virtualDirectoryKeyPrefix, directory,
 includeSubdirectories, metadataProvider)
```

```
tm.uploadDirectory(bucketName, virtualDirectoryKeyPrefix, directory,
 includeSubdirectories, metadataProvider, taggingProvider)
tm.uploadDirectory(bucketName, virtualDirectoryKeyPrefix, directory,
 includeSubdirectories, metadataProvider, taggingProvider, cannedAclProvider)

// V2
 ---------------------------------------------------------------------------------------------
UploadDirectoryRequest request = UploadDirectoryRequest.builder()
                                    .bucket(bucketName)
                                    .s3Prefix(virtualDirectoryKeyPrefix)
                                    .source(directory.toPath())
                                    .maxDepth(includeSubdirectories ? Integer.MAX_VALUE :
 1)
                                    .uploadFileRequestTransformer(builder -> {
                                        // 1.Replace `ObjectMetadataProvider`,
 `ObjectTaggingProvider`, and `ObjectCannedAclProvider` with an
                                          // `UploadFileRequestTransformer` that can
 combine the functionality of all three *Provider implementations.
                                          // 2. Convert your v1 `ObjectMetadata` to v2
 `PutObjectRequest` parameters.
                                          // 3. Convert your v1 `ObjectTagging` to v2
 `Tagging`.
                                          // 4. Convert your v1 `CannedAccessControlList`
 to v2 `ObjectCannedACL`.
                                    })
                                    .build();

DirectoryUpload directoryUpload = transferManager.uploadDirectory(request);
```

모델 객체 마이그레이션

AWS SDK for Java 2.x에서는 많은 `TransferManager` 모델 객체가 재설계되었으며 v1의 모델 객체에서 사용할 수 있는 여러 getter 및 setter 메서드는 더 이상 지원되지 않습니다.

v2에서는 `CompletableFuture<T>` 클래스를 사용하여 전송이 성공적으로 완료되거나 예외가 발생했을 때 작업을 수행합니다. 필요한 경우 `join()` 메서드를 사용하여 완료될 때까지 기다릴 수 있습니다.

주요 전송 객체

V1 클래스	V2 클래스	마이그레이션 상태
`TransferManager`	`S3TransferManager`	지원
`TransferManagerBuilder`	`S3TransferManager.Builder`	지원
`Transfer`	`Transfer`	지원
`AbortableTransfer`	`Transfer`	지원됨(별도 클래스 없음)
`Copy`	`Copy`	지원
`Download`	`FileDownload`	지원
`Upload`	`Upload` / `FileUpload`	지원
`MultipleFileDownload`	`DirectoryDownload`	지원
`MultipleFileUpload`	`DirectoryUpload`	지원

객체 지속성

V1 클래스	V2 클래스	마이그레이션 상태
`PersistableDownload`	`ResumableFileDownload`	지원
`PersistableUpload`	`ResumableFileUpload`	지원
`PersistableTransfer`	`ResumableTransfer`	지원
`PauseResult<T>`	직접 재개 가능한 객체	지원되지 않음

결과 객체

V1 클래스	V2 클래스	마이그레이션 상태
`CopyResult`	`CompletedCopy`	지원
`UploadResult`	`CompletedUpload`	지원

구성 객체

V1 클래스	V2 클래스	마이그레이션 상태
`TransferManagerConfiguration`	`MultipartConfiguration` (Amazon S3 클라이언트 기반)	지원
`TransferProgress`	`TransferProgress` + `TransferProgressSnapshot`	지원
`KeyFilter`	`DownloadFilter`	지원

지원되지 않는 객체

V1 클래스	V2 대안	마이그레이션 상태
`PauseStatus`	지원되지 않음	지원되지 않음
`UploadContext`	지원되지 않음	지원되지 않음
`ObjectCannedAclProvider`	`PutObjectRequest.builder().acl()`	지원되지 않음
`ObjectMetadataProvider`	`PutObjectRequest.builder().metadata()`	지원되지 않음

V1 클래스	V2 대안	마이그레이션 상태
`ObjectTaggingProvider`	`PutObjectRequest.builder().tagging()`	지원되지 않음
`PresignedUrlDownload`	지원되지 않음	지원되지 않음

TransferManagerBuilder 구성 마이그레이션

구성 변경

v2 Transfer Manager에 대해 설정해야 하는 구성 변경 사항은 사용하는 S3 클라이언트에 따라 다릅니다. AWS CRT 기반 S3 클라이언트 또는 표준 Java 기반 S3 비동기식 클라이언트 중에서 선택할 수 있습니다. 차이점에 대한 자세한 내용은 the section called “SDK의 S3 클라이언트” 주제를 참조하세요.

Use the AWS CRT-based S3 client

설정	v1	v2 - AWS CRT 기반 S3 클라이언트를 사용한 Transfer Manager
(빌더 구하기)	`TransferManagerBuilder tmBuilder = TransferManagerBuilder.standard();`	`S3TransferManager.Builder tmBuilder = S3TransferManager.builder();`
S3 클라이언트	`tmBuilder.withS3Client(...);` `tmBuilder.setS3Client(...);`	`tmBuilder.s3Client(...);`
실행자	`tmBuilder.withExecutorFactory(...);` `tmBuilder.setExecutorFactory(...);`	`tmBuilder.executor(...);`

설정	v1	v2 - AWS CRT 기반 S3 클라이언트를 사용한 Transfer Manager
스레드 풀 종료	`tmBuilder.withShutDownThreadPools(...);` `tmBuilder.setShutdownThreadPools(...);`	지원하지 않음. 제공된 실행기는 `S3TransferManager` 가 닫힐 때 종료되지 않습니다.

최소 업로드 파트 크기	`tmBuilder.withMinimumUploadPartSize(...);`
	`tmBuilder.setMinimumUploadPartSize(...);`
		`S3AsyncClient s3 = S3AsyncClient.crtBuilder().minimumPartSizeInBytes(...).build();`
		
		`tmBuilder.s3Client(s3);`

멀티파트 업로드 임계값	`tmBuilder.withMultipartUploadThreshold(...);`
	`tmBuilder.setMultipartUploadThreshold(...);`
		`S3AsyncClient s3 = S3AsyncClient.crtBuilder().thresholdInBytes(...).build();`
		
		`tmBuilder.s3Client(s3);`

설정	v1	v2 - AWS CRT 기반 S3 클라이언트를 사용한 Transfer Manager

최소 복사 파트 크기	`tmBuilder.withMultipartCopyPartSize(...);`
	`tmBuilder.setMultipartCopyPartSize(...);`
		`S3AsyncClient s3 = S3AsyncClient.crtBuilder().minimumPartSizeInBytes(...).build();`
		
		`tmBuilder.s3Client(s3);`

멀티파트 복사 임계값	`tmBuilder.withMultipartCopyThreshold(...);`
	`tmBuilder.setMultipartCopyThreshold(...);`
		`S3AsyncClient s3 = S3AsyncClient.crtBuilder().thresholdInBytes(...).build();`
		
		`tmBuilder.s3Client(s3);`

병렬 다운로드 비활성화	`tmBuilder.withDisableParallelDownloads(...);`
	`tmBuilder.setDisableParallelDownloads(...);`
		멀티파트 비활성화(기본값)으로 설정된 표준 Java 기반 S3 클라이언트를 Transfer Manager에 전달하여 병렬 다운로드를 비활성화합니다.
		
		`S3AsyncClient s3 = S3AsyncClient.builder().build();`
		
		`tmBuilder.s3Client(s3);`

설정	v1	v2 - AWS CRT 기반 S3 클라이언트를 사용한 Transfer Manager
항상 멀티파트 md5를 계산하세요.	`tmBuilder.withAlwaysCalculateMultipartMd5(...);` `tmBuilder.setAlwaysCalculateMultipartMd5(...);`	지원하지 않음.

Use Java-based S3 async client

설정	v1	v2 - Java 기반 S3 비동기식 클라이언트를 사용하는 Transfer Manager
(빌더 구하기)	`TransferManagerBuilder tmBuilder = TransferManagerBuilder.standard();`	`S3TransferManager.Builder tmBuilder = S3TransferManager.builder();`
S3 클라이언트	`tmBuilder.withS3Client(...);` `tmBuilder.setS3Client(...);`	`tmBuilder.s3Client(...);`
실행자	`tmBuilder.withExecutorFactory(...);` `tmBuilder.setExecutorFactory(...);`	`tmBuilder.executor(...);`
스레드 풀 종료	`tmBuilder.withShutDownThreadPools(...);`	지원하지 않음. 제공된 실행기는 `S3TransferManager` 가 닫힐 때 종료되지 않습니다.

설정	v1	v2 - Java 기반 S3 비동기식 클라이언트를 사용하는 Transfer Manager
	`tmBuilder.setShutdownThreadPools(...);`	

최소 업로드 파트 크기	`tmBuilder.withMinimumUploadPartSize(...);`
	`tmBuilder.setMinimumUploadPartSize(...);`
		`S3AsyncClient s3 = S3AsyncClient.builder()`
		`.multipartConfiguration(cfg -> cfg.minimumPartSizeInBytes(...)).build();`
		
		`tmBuilder.s3Client(s3);`

멀티파트 업로드 임계값	`tmBuilder.withMultipartUploadThreshold(...);`
	`tmBuilder.setMultipartUploadThreshold(...);`
		`S3AsyncClient s3 = S3AsyncClient.builder()`
		`.multipartConfiguration(cfg -> cfg.thresholdInBytes(...)).build();`
		
		`tmBuilder.s3Client(s3);`

설정	v1	v2 - Java 기반 S3 비동기식 클라이언트를 사용하는 Transfer Manager

최소 복사 파트 크기	`tmBuilder.withMultipartCopyPartSize(...);`
	`tmBuilder.setMultipartCopyPartSize(...);`
		`S3AsyncClient s3 = S3AsyncClient.builder()`
		`.multipartConfiguration(cfg ->`
		`cfg.minimumPartSizeInBytes(...)).build();`
		
		`tmBuilder.s3Client(s3);`

멀티파트 복사 임계값	`tmBuilder.withMultipartCopyThreshold(...);`
	`tmBuilder.setMultipartCopyThreshold(..);`
		`S3AsyncClient s3 = S3AsyncClient.builder()`
		`.multipartConfiguration(cfg ->`
		`cfg.thresholdInBytes(...)).build();`
		
		`tmBuilder.s3Client(s3);`

설정	v1	v2 - Java 기반 S3 비동기식 클라이언트를 사용하는 Transfer Manager

병렬 다운로드 비활성화	`tmBuilder.withDisableParallelDownloads(...);`
	`tmBuilder.setDisableParallelDownloads(...);`
		멀티파트 비활성화(기본값)으로 설정된 표준 Java 기반 S3 클라이언트를 Transfer Manager에 전달하여 병렬 다운로드를 비활성화합니다.
		`S3AsyncClient s3 = S3AsyncClient.builder().build();`
		`tmBuilder.s3Client(s3);`

항상 멀티파트 md5를 계산하세요.	`tmBuilder.withAlwaysCalculateMultipartMd5(...);` `tmBuilder.setAlwaysCalculateMultipartMd5(...);`	지원하지 않음.

동작 변경 사항

비동기식 운영

V1(차단):

```
Upload upload = transferManager.upload("amzn-s3-demo-bucket", "key", file);
upload.waitForCompletion(); // Blocks until complete
```

V2(비동기식):

```
FileUpload upload = transferManager.uploadFile(UploadFileRequest.builder()
    .putObjectRequest(PutObjectRequest.builder()
        .bucket("amzn-s3-demo-bucket")
```

```
        .key("key")
        .build())
    .source(file)
    .build());

CompletedFileUpload result = upload.completionFuture().join(); // Blocks until complete
// Or handle asynchronously:
upload.completionFuture().thenAccept(result -> {
    System.out.println("Upload completed: " + result.response().eTag());
});
```

오류 처리

V1: 하위 요청이 실패하면 디렉터리 전송이 완전히 실패합니다.

V2: 일부 하위 요청이 실패하더라도 디렉터리 전송이 성공적으로 완료됩니다. 다음과 같이 오류를 명시적으로 확인합니다.

```
DirectoryUpload directoryUpload = transferManager.uploadDirectory(request);
CompletedDirectoryUpload result = directoryUpload.completionFuture().join();

// Check for failed transfers
if (!result.failedTransfers().isEmpty()) {
    System.out.println("Some uploads failed:");
    result.failedTransfers().forEach(failed ->
        System.out.println("Failed: " + failed.exception().getMessage()));
}
```

바이트 범위 페치를 통한 병렬 다운로드

v2 SDK에서 자동 병렬 전송 기능이 활성화되면 S3 Transfer Manager v2는 바이트 범위 가져오기를 사용하여 객체의 특정 부분을 병렬로 검색(멀티파트 다운로드)합니다. v2를 사용하여 객체를 다운로드하는 방법은 객체가 처음 업로드된 방식에 따라 달라지지 않습니다. 모든 다운로드는 높은 처리량과 동시성의 이점을 누릴 수 있습니다.

이와 대조적으로 v1의 S3 Transfer Manager에서는 객체가 원래 업로드된 방식이 중요합니다. v1 Transfer Manager는 일부가 업로드된 것과 동일한 방식으로 객체의 일부를 검색합니다. 객체가 원래 단일 객체로 업로드된 경우 v1 Transfer Manager는 하위 요청을 사용하여 다운로드 프로세스를 가속화할 수 없습니다.

버전 1에서 버전 2로 Amazon S3 URI 구문 분석에 관한 변경 사항

이 주제에서는 버전 1(v1)에서 버전 2(v2)로의 Amazon S3 URI 구문 분석의 변경 사항에 대해 자세히 설명합니다.

높은 수준의 변경 사항

v1에서 S3 URI 구문 분석을 시작하려면 생성자를 사용하여 `AmazonS3URI`를 인스턴스화합니다. v2에서는 `S3Utilities`의 인스턴스에서 `parseUri()`를 호출하여 `S3URI`를 반환합니다.

변경 사항	v1	v2

Maven 종속성	`<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>s3</artifactId>`
	`  </dependency>`
	`</dependencies>`
		`<dependencyManagement>`
		`  <dependencies>`
		`  <dependency>`
		`  <groupId>software.amazon.awssdk</groupId>`
		`  <artifactId>bom</artifactId>`
		`  <version>2.27.21[2]</version>`
		`  <type>pom</type>`
		`  <scope>import</scope>`
		`  </dependency>`
		`  </dependencies>`
		`</dependencyManagement>`
		`<dependencies>`
		`  <dependency>`
		`  <groupId>software.amazon.awssdk</groupId>`
		`  <artifactId>s3</artifactId>`
		`  </dependency>`
		`</dependencies>`

변경 사항	v1	v2
패키지 이름	`com.amazonaws.services.s3`	`software.amazon.awssdk.services.s3`
클래스 이름	`AmazonS3URI`	`S3URI`

[1] 최신 버전. [2] 최신 버전.

API 변경 사항

동작	v1	v2

S3 URI를 구문 분석합니다.	`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);`

S3 버킷에서 버킷 이름을 검색합니다.	`String bucket = s3Uri.getBucket();`	`Optional<String> bucket = s3Uri.bucket();`
키를 검색합니다.	`String key = s3Uri.getKey();`	`Optional<String> key = s3Uri.key();`

리전을 검색합니다.	`String region = s3Uri.getRegion();`	`Optional<Region> region = s3Uri.region();`
		
		`String region;`
		`if (s3Uri.region().isPresent()) {`
		`region = s3Uri.region().get().id();`

동작	v1	v2
		`}`
S3 URI가 경로형인지 검색합니다.	`boolean isPathStyle = s3Uri.isPathStyle();`	`boolean isPathStyle = s3Uri.isPathStyle();`
버전 ID를 검색합니다.	`String versionId = s3Uri.getVersionId();`	`Optional<String> versionId = s3Uri.firstMatchingRawQueryParameter("versionId");`
쿼리 파라미터를 검색합니다.	N/A	`Map<String, List<String>> queryParams = s3Uri.rawQueryParameters();`

동작 변경 사항

URL 인코딩

v1은 플래그를 전달하여 URI를 URL로 인코딩할지 여부를 지정하는 옵션을 제공합니다. 기본값은 `true`입니다.

v2에서는 URL 인코딩이 지원되지 않습니다. 예약되었거나 안전하지 않은 문자가 있는 객체 키 또는 쿼리 파라미터로 작업하는 경우 URL 인코딩해야 합니다. 예를 들어 공백(" ")을 `%20`으로 바꿔야 합니다.

S3 이벤트 알림 API를 버전 1에서 버전 2로 변경

이 주제에서는 S3 이벤트 알림 API를 AWS SDK for Java의 버전 1.x(v1)에서 버전 2 .x(v2)로 변경에 대해 자세히 설명합니다.

높은 수준의 변경 사항

구문 변경 사항

V1은 `EventNotificationRecord` 유형 및 속성에 정적 내부 클래스를 사용하는 반면, v2는 `EventNotificationRecord` 유형에 대해 별도의 퍼블릭 클래스를 사용합니다.

이름 지정 규칙 변경 사항

V1에서 속성 클래스 이름에는 접미사 엔터티가 포함되는 반면, V2에서는 보다 간단한 이름 지정을 위해 이 접미사를 생략합니다. 예를 들어 eventDataEntity가 아닌 eventData입니다.

종속성, 패키지 및 클래스 이름의 변경 사항

V1에서 S3 이벤트 알림 API 클래스는 S3 모듈(artifactId `aws-java-sdk-s3`)과 함께 사용하여 전이적으로 가져옵니다. 그러나 V2에서는 `s3-event-notifications` 아티팩트에 대한 종속성을 추가해야 합니다.

변경 사항	v1	v2
Maven 종속성	`<dependencyManagement> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> 1.X.X</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency>`	`<dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> 2.X.X[1]</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId>`

변경 사항	v1	v2
	<groupId>com.amazonaws</groupId> <artifactId>aws-java-sdk-s3</artifactId> </dependency> </dependencies>	<artifactId>s3-event-notifications</artifactId> </dependency> </dependencies>
패키지 이름	com.amazonaws.services.s3.event	software.amazon.awssdk.eventnotifications.s3.model

변경 사항	v1	v2
클래스 이름	S3EventNotification S3EventNotification.S3EventNotificationRecord S3EventNotification.GlacierEventDataEntity S3EventNotification.IntelligentTieringEventDataEntity S3EventNotification.LifecycleEventDataEntity S3EventNotification.ReplicationEventDataEntity S3EventNotification.RequestParametersEntity S3EventNotification.ResponseElementsEntity S3EventNotification.RestoreEventDataEntity S3EventNotification.S3BucketEntity S3EventNotification.S3Entity S3EventNotification.S3ObjectEntity S3EventNotification.TransitionEventDataEntity	S3EventNotification S3EventNotificationRecord GlacierEventData IntelligentTieringEventData LifecycleEventData ReplicationEventData RequestParameters ResponseElements RestoreEventData S3Bucket S3 'S3Object': TransitionEventData UserIdentity

변경 사항	v1	v2
	S3EventNotification.UserIdentityEntity	

[1] 최신 버전.

API 변경 사항

S3EventNotification에 대한 JSON 및 역방향

사용 사례	v1	v2
JSON 문자열에서 S3EventNotification 만들기	`S3EventNotification notification = S3EventNotification.parseJson(message.body());`	`S3EventNotification notification = S3EventNotification.fromJson(message.body());`
JSON 문자열로 S3EventNotification 변환	`String json = notification.toJson();`	`String json = notification.toJson();`

S3EventNotification의 액세스 속성

사용 사례	v1	v2
알림에서 레코드 검색	`List<S3EventNotification.S3EventNotificationRecord> records = notifcation.getRecords();`	`List<S3EventNotificationRecord> records = notification.getRecords();`
레코드 목록에서 레코드 검색	`S3EventNotification.S3EventNotificationRecord record =`	`S3EventNotificationRecord record =`

사용 사례	v1	v2
	`records.stream().findAny().get();`	`records.stream().findAny().get();`
Glacier 이벤트 데이터 검색	`S3EventNotification.GlacierEventDataEntity glacierEventData = record.getGlacierEventData();`	`GlacierEventData glacierEventData = record.getGlacierEventData();`
Glacier 이벤트에서 복원 이벤트 데이터 검색	`S3EventNotification.RestoreEventDataEntity restoreEventData = glacierEventData.getRestoreEventDataEntity();`	`RestoreEventData restoreEventData = glacierEventData.getRestoreEventData();`
요청 파라미터 검색	`S3EventNotification.RequestParametersEntity requestParameters = record.getRequestParameters();`	`RequestParameters requestParameters = record.getRequestParameters();`
지능형 계층화 이벤트 데이터 검색	`S3EventNotification.IntelligentTieringEventDataEntity tieringEventData = record.getIntelligentTieringEventData();`	`IntelligentTieringEventData intelligentTieringEventData = record.getIntelligentTieringEventData();`
수명 주기 이벤트 데이터 검색	`S3EventNotification.LifecycleEventDataEntity lifecycleEventData = record.getLifecycleEventData();`	`LifecycleEventData lifecycleEventData = record.getLifecycleEventData();`

사용 사례	v1	v2
이벤트 이름을 열거형으로 검색	`S3Event eventNameAsEnum = record.getEventNameAsEnum();`	`//getEventNameAsEnum does not exist; use 'getEventName()' String eventName = record.getEventName();`
복제 이벤트 데이터 검색	`S3EventNotification.ReplicationEventDataEntity replicationEntity = record.getReplicationEventDataEntity();`	`ReplicationEventData replicationEventData = record.getReplicationEventData();`
S3 버킷 및 객체 정보 검색	`S3EventNotification.S3Entity s3 = record.getS3();`	`S3 s3 = record.getS3();`
사용자 자격 증명 정보 검색	`S3EventNotification.UserIdentityEntity userIdentity = record.getUserIdentity();`	`UserIdentity userIdentity = record.getUserIdentity();`
응답 요소 검색	`S3EventNotification.ResponseElementsEntity responseElements = record.getResponseElements();`	`ResponseElements responseElements = record.getResponseElements();`

aws-lambda-java-events 라이브러리 버전 마이그레이션

aws-lambda-java-events를 사용하여 Lambda 함수 내에서 S3 알림 이벤트를 사용하는 경우 최신 3.x.x 버전으로 업그레이드하는 것이 좋습니다. 최신 버전은 S3 이벤트 알림 API에서 AWS SDK for Java 1.x에 대한 모든 종속성을 제거합니다.

`aws-lambda-java-events` 라이브러리와 SDK for Java 2.x 간의 S3 이벤트 알림 처리 차이점에 대한 자세한 내용은 the section called “S3 이벤트 알림 처리 옵션” 섹션을 참조하세요.

프로필 파일 변경 사항

AWS SDK for Java 2.x는 `~/.aws/config` 및 `~/.aws/credentials`에서 프로필 정의를 구문 분석하여 AWS CLI가 파일을 구문 분석하는 방법을 더 자세히 에뮬레이션합니다.

SDK for Java 2.x:

- `$HOME`, `$USERPROFILE`(Windows만 해당), `$HOMEDRIVE`, `$HOMEPATH`(Windows만 해당) 및 `user.home` 시스템 속성을 순서대로 확인하여 경로 시작 시 파일 시스템의 기본 경로 구분자가 뒤따르는 `~/` 또는 `~` 코드를 해결합니다.
- `AWS_CREDENTIAL_PROFILES_FILE` 대신 `AWS_SHARED_CREDENTIALS_FILE` 환경 변수를 찾습니다.
- 프로필 이름의 시작 부분에 `profile` 단어 없이 구성 파일의 프로필 정의를 자동으로 삭제합니다.
- 영숫자, 밑줄 또는 대시 문자로 구성되지 않은 프로필 정의를 자동으로 삭제합니다(구성 파일에 대해 선행 `profile` 단어를 제거한 후).
- 동일한 파일 내에 복제된 프로필 정의의 설정을 병합합니다.
- 구성 파일과 자격 증명 파일 모두에 중복된 프로필 정의의 설정을 병합합니다.
- 동일한 파일에서 `[profile foo]` 및 `[foo]`가 모두 발견되면 설정을 병합하지 않습니다.
- 구성 파일에서 `[profile foo]` 및 `[foo]`를 모두 찾을 경우 `[profile foo]`의 설정을 사용합니다.
- 동일한 파일 및 프로필에서 마지막으로 복제된 설정의 값을 사용합니다.
- 주석을 정의하기 위해 `;` 및 `#` 기호를 모두 인식합니다.
- 문자가 닫는 대괄호 옆에 있더라도 프로필 정의에서 `;` 및 `#` 기호를 인식하여 주석을 정의합니다.
- 설정 값 앞에 공백이 있는 경우에만 주석을 정의하기 위해 `;` 및 `#` 기호를 인식합니다.
- 앞에 공백이 없는 경우 설정 값에서 `;` 및 `#`, 다음 모든 콘텐츠를 인식합니다.
- 역할 기반 자격 증명을 우선순위가 가장 높은 자격 증명으로 간주합니다. 사용자가 `role_arn` 속성을 지정하는 경우 2.x SDK는 항상 역할 기반 자격 증명을 사용합니다.
- 세션 기반 자격 증명을 차순위 자격 증명으로 간주합니다. 역할 기반 자격 증명을 사용하지 않고 사용자가 `aws_access_key_id` 및 `aws_session_token` 속성을 지정하는 경우 2.x SDK는 항상 세션 기반 자격 증명을 사용합니다.

- 역할 기반 및 세션 기반 자격 증명을 사용하지 않고 사용자가 aws_access_key_id 속성을 지정한 경우 기본 자격 증명을 사용합니다.

환경 변수 및 시스템 속성 변경

1.x 환경 변수	1.x 시스템 속성	2.x 환경 변수	2.x 시스템 속성
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
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.ec2MetadataService	AWS_EC2_METADATA_SERVICE_ENDPOINT	aws.ec2MetadataServiceEndpoint

1.x 환경 변수	1.x 시스템 속성	2.x 환경 변수	2.x 시스템 속성
	Endpoint0 verride		
AWS_CONTA INER_CRED ENTIALS_R ELATIVE_URI		AWS_CONTA INER_CRED ENTIALS_R ELATIVE_URI	aws.conta inerCrede ntialsPath
AWS_CONTA INER_CRED ENTIALS_F ULL_URI		AWS_CONTA INER_CRED ENTIALS_F ULL_URI	aws.conta inerCrede ntialsFullUri
AWS_CONTA INER_AUTH ORIZATION _TOKEN		AWS_CONTA INER_AUTH ORIZATION _TOKEN	aws.conta inerAutho rizationToken
AWS_CBOR_ DISABLED	com.amazo naws.sdk. disableCbor	CBOR_ENABLED	aws.cborEnabled
AWS_ION_B INARY_DISABLE	com.amazo naws.sdk. disableIo nBinary	BINARY_IO N_ENABLED	aws.binar yIonEnabled
AWS_EXECU TION_ENV		AWS_EXECU TION_ENV	aws.execu tionEnvir onment
	com.amazo naws.sdk. disableCe rtChecking	비지원(요청 기능)	비지원(요청 기능)

1.x 환경 변수	1.x 시스템 속성	2.x 환경 변수	2.x 시스템 속성
	com.amazonaws.sdk.enableDefaultMetrics	지원되지 않음	지원되지 않음
	com.amazonaws.sdk.enableThrottledRetry	지원되지 않음	지원되지 않음
	com.amazonaws.regions.RegionUtils.fileOverride	비지원(요청 기능)	비지원(요청 기능)
	com.amazonaws.regions.RegionUtils.disableRemote	비지원(요청 기능)	비지원(요청 기능)
	com.amazonaws.services.s3.disableImplicitGlobalClients	비지원(요청 기능)	비지원(요청 기능)
	com.amazonaws.sdk.enableInRegionOptimizedMode	비지원(요청 기능)	비지원(요청 기능)

웨이터를 버전 1에서 버전 2로 변경

이 주제에서는 버전 1(v1)에서 버전 2(v2)로 변환 시 웨이터 기능의 변경 사항을 자세히 설명합니다.

다음 표에서는 DynamoDB 웨이터의 차이점을 구체적으로 보여줍니다. 다른 서비스의 웨이터는 동일한 패턴을 따릅니다.

높은 수준의 변경 사항

웨이터 클래스는 서비스와 동일한 Maven 아티팩트에 있습니다.

변경 사항	v1	v2
Maven 종속성	<dependencyManagement> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> *1.12.680*[1]</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>dynamodb</artif actId> </dependency> </dependencies>	<dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> *2.27.10*[2]</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>dynamodb</artif actId> </dependency> </dependencies>

변경 사항	v1	v2
패키지 이름	`com.amazonaws.services.dynamodbv2.waiters`	`software.amazon.awssdk.services.dynamodb.waiters`
클래스 이름	AmazonDynamoDBWaiters	• 동기식: DynamoDbWaiter • 비동기식: DynamoDbAsyncWaiter

[1] 최신 버전. [2] 최신 버전.

API 변경 사항

변경 사항	v1	v2
웨이터 만들기	`AmazonDynamoDB client = AmazonDynamoDBClientBuilder .standard().build(); AmazonDynamoDBWaiters waiter = client.waiters();`	동기식: `DynamoDbClient client = DynamoDbClient.create(); DynamoDbWaiter waiter = client.waiter();` 비동기식: `DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create(); DynamoDbAsyncWaiter waiter = asyncClient.waiter();`
테이블이 존재할 때까지 대기	동기식: `waiter.tableExists()`	동기식:

변경 사항	v1	v2

	.run(new WaiterParameters<>(
	new DescribeTableRequest(tableName)));
	
	비동기식:
	
	waiter.tableExists()
	.runAsync(new WaiterParameters()
	.withRequest(new DescribeTableRequest(tableName)),
	new WaiterHandler() {
	@Override
	public void onWaitSuccess(
	AmazonWebServiceRequest amazonWebServiceRequest) {
	System.out.println("Table creation succeeded");
	}
	@Override
	public void onWaitFailure(Exception e) {
	e.printStackTrace();
	}
	}).get();
		WaiterResponse<DescribeTableResponse> waiterResponse =
		waiter.waitUntilTableExists(
		r -> r.tableName("myTable"));
		waiterResponse.matched().response()
		.ifPresent(System.out::println);
		
		비동기식:
		
		waiter.waitUntilTableExists(r ->
		r.tableName(tableName))
		.whenComplete((r, t) -> {
		if (t != null) {
		t.printStackTrace();
		} else {
		System.out.println(
		"Table creation succeeded");
		}
		}).join();

구성 변경

변경 사항	v1	v2

폴링 전략(최대 시도 횟수 및 고정 지연)	`MaxAttemptsRetryStrategy maxAttemptsRetryStrategy = new MaxAttemptsRetryStrategy(10);`
	
	`FixedDelayStrategy fixedDelayStrategy = new FixedDelayStrategy(3);`
	
	`PollingStrategy pollingStrategy = new PollingStrategy(maxAttemptsRetryStrategy, fixedDelayStrategy);`
	
	`waiter.tableExists().run( new WaiterParameters<>( new DescribeTableRequest(tableName)), pollingStrategy);`
		`FixedDelayBackoffStrategy fixedDelayBackoffStrategy = FixedDelayBackoffStrategy .create(Duration.ofSeconds(3));`
		
		`waiter.waitUntilTableExists(r -> r.tableName(tableName), c -> c.maxAttempts(10) .backoffStrategy(fixedDelayBackoffStrategy));`

EC2 메타데이터 유틸리티가 버전 1에서 버전 2로 변경

이 항목에서는 버전 1(v1)에서 버전 2(v2)로 변경된 SDK for Java Amazon Elastic Compute Cloud(EC2) 메타데이터 유틸리티의 변경 사항에 대해 자세히 설명합니다.

높은 수준의 변경 사항

변경	v1	v2
Maven 종속성	<dependencyManagement> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> 1.12.587[1]</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-co re</artifactId> </dependency> </dependencies>	<dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> 2.27.21[2]</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>imds</artifactId> </dependency> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>apache-client [3]</ artifactId> </dependency> </dependencies>

변경	v1	v2
패키지 이름	`com.amazonaws.util`	`software.amazon.awssdk.imds`

인스턴스화 접근 방식	인스턴스화 없이 정적 유틸리티 메서드 사용:
	`String localHostName = EC2MetadataUtils.getLocalHostName();`
		정적 팩토리 메서드 사용:
		`Ec2MetadataClient client = Ec2MetadataClient.create();`
		또는 빌더 접근 방식 사용:
		`Ec2MetadataClient client = Ec2MetadataClient.builder() .endpointMode(EndpointMode.IPV6) .build();`

클라이언트 유형	동기 전용 유틸리티 메서드: `EC2MetadataUtils`	동기식: `Ec2MetadataClient` 비동기식: `Ec2MetadataAsyncClient`

[1] 최신 버전. [2] 최신 버전.

[3]v2에 대한 `apache-client` 모듈 선언에 주목합니다. EC2 메타데이터 유틸리티 V2를 사용하려면 동기식 메타데이터 클라이언트의 경우 `SdkHttpClient` 인터페이스를 구현해야 하고, 비동기식 메타데이터 클라이언트의 경우 `SdkAsyncHttpClient` 인터페이스를 구현해야 합니다. ??? 섹션에는 사용할 수 있는 HTTP 클라이언트 목록이 표시됩니다.

메타데이터 요청

v1에서는 파라미터를 허용하지 않는 정적 메서드를 사용하여 EC2 리소스에 대한 메타데이터를 요청합니다. 이와는 대조적으로 v2에서는 EC2 리소스의 경로를 파라미터로 지정해야 합니다. 다음 표는 서로 다른 접근 방식을 보여줍니다.

v1	v2
`String userMetaData = EC2Metadata Utils.getUserData();`	`Ec2MetadataClient client = Ec2Metadata Client.create(); Ec2MetadataResponse response = client.get("/latest/user-data"); String userMetaData = response.asString();`

인스턴스 메타데이터 범주를 참조하여 메타데이터를 요청하기 위해 제공해야 하는 경로를 찾으세요.

> **Note**
>
> v2에서 인스턴스 메타데이터 클라이언트를 사용하는 경우, 메타데이터를 검색하는 모든 요청에 동일한 클라이언트를 사용해야 합니다.

동작 변경 사항

JSON 데이터

EC2에서 로컬로 실행되는 인스턴스 메타데이터 서비스(IMDS)는 일부 메타데이터를 JSON 형식의 문자열로 반환합니다. 그러한 예로 인스턴스 ID 문서의 동적 메타데이터를 들 수 있습니다.

v1 API에는 인스턴스 ID 메타데이터의 각 부분에 대한 별도의 메서드가 포함되어 있는 반면, v2 API는 JSON 문자열을 직접 반환합니다. JSON 문자열로 작업하려면 문서 API를 사용하여 응답을 구문 분석하고 JSON 구조를 탐색할 수 있습니다.

다음 표는 v1과 v2에서 인스턴스 ID 문서의 메타데이터를 검색하는 방법을 비교한 것입니다.

사용 사례:	v1	v2
리전 검색	`InstanceInfo instanceInfo = EC2MetadataUtils.getInstanceInfo();`	`Ec2MetadataResponse response = client.get("/latest/dynamic/instance-identity/document");`

사용 사례:	v1	v2
	`String region = instanceInfo.getRegion();`	`Document instanceInfo = response.asDocument(); String region = instanceInfo.asMap().get("region").asString();`
인스턴스 ID 검색	`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();`
인스턴스 유형 검색	`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().get("instanceType").asString();`

엔드포인트 해상도 차이

다음 표는 SDK가 엔드포인트를 IMDS로 확인하기 위해 확인하는 위치를 보여줍니다. 위치는 내림차순으로 나열됩니다.

v1	v2
시스템 속성: `com.amazonaws.sdk.ec2MetadataServiceEndpointOverride`	클라이언트 빌더 구성 메서드: `endpoint(...)`
환경 변수: `AWS_EC2_METADATA_SERVICE_ENDPOINT`	시스템 속성: `aws.ec2MetadataServiceEndpoint`
기본 값: `http://169.254.169.254`	구성 파일: `ec2_metadata_service_endpoint` 설정이 포함된 `~.aws/config`
	확인된 `endpoint-mode` 관련된 값
	기본값: `http://169.254.169.254`

v2의 엔드포인트 해상도

빌더를 사용하여 엔드포인트를 명시적으로 설정하면 해당 엔드포인트 값이 다른 모든 설정보다 우선합니다. 다음 코드가 실행될 때 `aws.ec2MetadataServiceEndpoint` 시스템 속성 및 구성 파일 `ec2_metadata_service_endpoint` 설정이 있는 경우 무시됩니다.

```
Ec2MetadataClient client = Ec2MetadataClient
  .builder()
  .endpoint(URI.create("endpoint.to.use"))
  .build();
```

엔드포인트 모드

v2에서는 엔드포인트 모드를 지정하여 IPv4 또는 IPv6의 기본 엔드포인트 값을 사용하도록 메타데이터 클라이언트를 구성할 수 있습니다. v1에는 엔드포인트 모드를 사용할 수 없습니다. IPv4에 사용되는 기본값은 `http://169.254.169.254`, IPv6의 경우 `http://[fd00:ec2::254]`입니다.

다음 표는 우선 순위가 내려가는 순서대로 엔드포인트 모드를 설정할 수 있는 다양한 방법을 보여줍니다.

		가능한 값
클라이언트 빌더 구성 메서드: `endpointMode(...)`	`Ec2MetadataClient client = Ec2MetadataClient.builder().endpointMode(EndpointMode.IPV4).build();`	`EndpointMode.IPV4` , `EndpointMode.IPV6`
시스템 속성	`aws.ec2MetadataServiceEndpointMode`	IPv4, IPv6(대/소문자를 구분하지 않음)
Config 파일: `~.aws/config`	`ec2_metadata_service_endpoint` 설정	IPv4, IPv6(대/소문자를 구분하지 않음)
이전 방법으로는 지정되지 않음	IPv4가 사용됨	

SDK가 v2에서 **`endpoint`** 또는 **`endpoint-mode`**를 확인하는 방법

1. SDK는 클라이언트 빌더의 코드에서 설정한 값을 사용하며 외부 설정은 무시합니다. 클라이언트 빌더에서 `endpoint`와 `endpointMode`가 모두 호출되면 SDK는 예외를 던지므로, 어떤 메서드를 사용하든 엔드포인트 값을 사용합니다.
2. 코드에서 값을 설정하지 않으면 SDK는 외부 구성에서 먼저 시스템 속성을 찾은 다음 구성 파일의 설정을 찾습니다.
   a. SDK는 먼저 엔드포인트 값을 확인합니다. 값이 발견되면 해당 값이 사용됩니다.
   b. 여전히 값을 찾지 못한 경우 SDK는 엔드포인트 모드 설정을 찾습니다.
3. 마지막으로, SDK가 외부 설정을 찾지 못하고 코드에서 메타데이터 클라이언트를 구성하지 않은 경우 SDK는 `http://169.254.169.254`의 IPv4 값을 사용합니다.

IMDSv2

Amazon EC2는 인스턴스 메타데이터에 액세스하는 두 가지 접근 방식을 정의합니다.

- 인스턴스 메타데이터 서비스 버전 1(IMDSv1) - 요청/응답 방식

- 인스턴스 메타데이터 서비스 버전 2(IMDSv2) - 세션 지향 방식

다음 표는 Java SDK와 IMDS의 작동 방식을 비교한 것입니다.

v1	v2
IMDSv2가 기본적으로 사용됨	항상 IMDSv2를 사용함
각 요청에 대해 세션 토큰을 가져오려고 시도하고 세션 토큰을 가져오지 못하면 IMDSv1으로 돌아갑니다.	여러 요청에 재사용되는 세션 토큰을 내부 캐시에 보관합니다.

SDK for Java 2.x는 IMDSv2만 지원하며 IMDSv1로 돌아가지 않습니다.

구성 차이점

다음 표에는 다양한 구성 옵션이 나와 있습니다.

구성	v1	v2
재시도	구성을 사용할 수 없음	빌더 메서드 `retryPolicy(...)` 를 통해 구성할 수 있습니다.
HTTP	연결 시간 제한은 `AWS_METADATA_SERVICE_TIMEOUT` 환경 변수를 통해 구성할 수 있습니다. 기본값은 1초입니다.	빌더 메서드인 `httpClient(...)` 에 HTTP 클라이언트를 전달하여 구성할 수 있습니다. HTTP 클라이언트의 기본 연결 시간 제한은 2초입니다.

예제 v2 HTTP 구성

다음 예제는 메타데이터 클라이언트를 구성하는 방법을 보여줍니다. 이 예제에서는 연결 시간 제한을 구성하고 Apache HTTP 클라이언트를 사용합니다.

```
SdkHttpClient httpClient = ApacheHttpClient.builder()
    .connectionTimeout(Duration.ofSeconds(1))
```

```
    .build();

Ec2MetadataClient imdsClient = Ec2MetadataClient.builder()
    .httpClient(httpClient)
    .build();
```

버전 1에서 버전 2로 Amazon CloudFront 사전 서명 변경

이 주제에서는 버전 1(v1)에서 버전 2(v2)로의 Amazon CloudFront 변경 사항에 대해 자세히 설명합니다.

높은 수준의 변경 사항

변경 사항	v1	v2
Maven 종속성	<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>cloudfront</artifactId>	<dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.awssdk</groupId> <artifactId>bom</artifactId> <version> *2.27.21*[2]</version> <type>pom</type> <scope>import</scope> </dependency> </dependencies> </dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.awssdk</groupId> <artifactId>cloudfront</artifactId> </dependency>

변경 사항	v1	v2
	</dependency> </dependencies>	</dependencies>
패키지 이름	com.amazonaws.services.cloudfront	software.amazon.awssdk.services.cloudfront
클래스 이름	CloudFrontUrlSigner CloudFrontCookieSigner	CloudFrontUtilities SignedUrl CannedSignerRequest CustomSignerRequest

[1] 최신 버전. [2] 최신 버전.

API 변경 사항

동작	v1	v2
미리 준비된 요청 구축	인수는 API로 직접 전달됩니다.	`CannedSignerRequest cannedRequest = CannedSignerRequest.builder() .resourceUrl(resourceUrl) .privateKey(privateKey) .keyPairId(keyPairId) .expirationDate(expirationDate) .build();`

동작	v1	v2
사용자 지정 요청 구축	인수는 API로 직접 전달됩니다.	`CustomSignerRequest customRequest = CustomSignerRequest.builder() .resourceUrl(resourceUrl) .resourceUrlPattern(resourceUrlPattern) .privateKey(keyFile) .keyPairId(keyPairId) .expirationDate(expirationDate) .activeDate(activeDate) .ipRange(ipRange) .build();`
서명된 URL 생성(미리 준비됨)	`String signedUrl = CloudFrontUrlSigner.getSignedURLWithCannedPolicy( resourceUrl, keyPairId, privateKey, expirationDate);`	`CloudFrontUtilities cloudFrontUtilities = CloudFrontUtilities.create(); SignedUrl signedUrl = cloudFrontUtilities.getSignedUrlWithCannedPolicy(cannedRequest); String url = signedUrl.url();`

동작	v1	v2

서명된 쿠키 생성(사용자 지정)	`CookiesForCustomPolicy cookies = CloudFrontCookieSigner.getCookiesForCustomPolicy( resourceUrl, privateKey, keyPairId, expirationDate, activeDate, ipRange);`	`CloudFrontUtilities cloudFrontUtilities = CloudFrontUtilities.create();`
		
		`CookiesForCustomPolicy cookies = cloudFrontUtilities.getCookiesForCustomPolicy(customRequest);`

v2의 리팩터링된 쿠키 헤더

Java v1에서 Java SDK는 쿠키 헤더를 `Map.Entry<String, String>`으로 제공합니다.

```
Map.Entry<String, String> signatureMap = cookies.getSignature();
String signatureKey = signatureMap.getKey(); // "CloudFront-Signature"
String signatureValue = signatureMap.getValue(); // "[SIGNATURE_VALUE]"
```

Java v2 SDK는 전체 헤더를 단일 `String`으로 제공합니다.

```
String signatureHeaderValue = cookies.signatureHeaderValue(); // "CloudFront-
Signature=[SIGNATURE_VALUE]"
```

IAM 정책 빌더 API를 버전 1에서 버전 2로 변경

이 주제에서는 버전 1(v1)에서 버전 2(v2)로의 IAM 정책 빌더 API의 변경 사항에 대해 자세히 설명합니다.

높은 수준의 변경 사항

변경 사항	v1	v2
Maven 종속성	`<dependencyManagement> <dependencies>`	`<dependencyManagement> <dependencies>`

변경 사항	v1	v2
	<dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> *1.12.587*[1]</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-co re</artifactId> </dependency> </dependencies>	<dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> *2.27.21*[2]</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>iam-policy-buil der</artifactId> </dependency> </dependencies>
패키지 이름	`com.amazonaws.auth.policy`	`software.amazon.awssdk.policybuilder.iam`

변경 사항	v1	v2
클래스 이름	정책 문 • Statement.Effect • IdentiyManagementActions • 리소스 • 보안 주체 • Condition	IamPolicy IamStatement • IamEffect • IamAction • IamResource • IamPrincipal • IamCondition • IamConditionOperator • IamConditionKey

[1] 최신 버전. [2] 최신 버전.

API 변경 사항

설정	v1	v2
정책 인스턴스화	`Policy policy = new Policy();`	`IamPolicy.Builder policyBuilder = IamPolicy.builder(); ... IamPolicy policy = policyBuilder.build();`
세트 ID	`policy.withtId(...); policy.setId(...);`	`policyBuilder.id(...);`
세트 버전	해당 사항 없음 - 2012-10-17 의 기본 버전을 사용합니다.	`policyBuilder.version(...);`
CREATE 문	`Statement statement =`	`IamStatement statement =`

설정	v1	v2
	`new Statement (Effect.Allow) .withActions(...) .withConditions(...) .withId(...) .withPrincipals(...) .withResources(...);`	`IamStatement.Builder() .effect(IamEffect.ALLOW) .actions(...) .notActions(...) .conditions(...) .sid(...) .principals(...) .notPrincipals(...) .resources(...) .notResources(...) .build()`
SET 문	`policy.withStatements(statement); policy.setStatements(statement);`	`policyBuilder.addStatement(statement);`

문 구축에서의 차이점

작업

v1

v1 SDK에는 정책 문의 *Action* 요소를 나타내는 서비스 작업에 대한 enum 유형이 포함됩니다. 다음 enum 유형은 몇 가지 예제입니다.

- IdentityManagementActions
- DynamoDBv2Actions
- SQSActions

다음 예제에서는 SQSActions에 대한 SendMessage 상수를 보여줍니다.

```
Action action = SQSActions.SendMessage;
```

v1의 문에 대해 *NotAction* 요소를 지정할 수 없습니다.

v2

v2에서 IamAction 인터페이스는 모든 작업을 나타냅니다. 서비스별 작업 요소를 지정하려면 다음 코드와 같이 문자열을 create 메서드에 전달합니다.

```
IamAction action = IamAction.create("sqs:SendMessage");
```

다음 코드와 같이 v2가 있는 문에 대해 *NotAction*을 지정할 수 있습니다.

```
IamAction action = IamAction.create("sqs:SendMessage");
IamStatement.builder().addNotAction(action);
```

조건

v1

문 조건을 나타내기 위해 v1 SDK는의 Condition의 하위 클래스를 사용합니다.

- ArnCondition
- BooleanCondition
- DateCondition
- IpAddressCondition
- NumericCondition
- StringCondition

각 Condition 하위 클래스는 조건을 정의하는 데 도움이 되는 비교 enum 유형을 정의합니다. 예를 들어 다음은 조건에 대한 유사하지 않은 문자열 비교를 보여줍니다.

```
Condition condition = new StringCondition(StringComparisonType.StringNotLike, "key",
 "value");
```

v2

v2에서는 IamCondition을 사용하여 정책 설명에 대한 조건을 구축하고 모든 유형에 대해 enums가 포함된IamConditionOperator를 제공합니다.

```
IamCondition condition = IamCondition.create(IamConditionOperator.STRING_NOT_LIKE,
 "key", "value");
```

리소스

v1

정책 문의 *Resource* 요소는 SDK의 Resource 클래스로 표시됩니다. 생성자에서 ARN을 문자열로 제공합니다. 다음 하위 클래스는 편의 생성자를 제공합니다.

- S3BucketResource
- S3ObjectResource
- SQSQueueResource

v1에서는 다음 문과 같이 withIsNotType 메서드를 호출하여 Resource에 대한 *NotResource* 요소를 지정할 수 있습니다.

```
Resource resource = new Resource("arn:aws:s3:::amzn-s3-demo-
bucket").withIsNotType(true);
```

v2

v2에서는 ARN을 IamResource.create 메서드에 전달하여 *Resource* 요소를 만듭니다.

```
IamResource resource = IamResource.create("arn:aws:s3:::amzn-s3-demo-bucket");
```

IamResource는 다음 코드 조각과 같이 *NotResource* 요소로 설정할 수 있습니다.

```
IamResource resource = IamResource.create("arn:aws:s3:::amzn-s3-demo-bucket");
IamStatement.builder().addNotResource(resource);
```

IamResource.ALL은 모든 리소스를 나타냅니다.

보안 주체

v1

v1 SDK는 모든 멤버를 포함하는 보안 주체 유형을 나타내는 다음 `Principal` 클래스를 제공합니다.

- `AllUsers`
- `AllServices`
- `AllWebProviders`
- `All`

문에 *`NotPrincipal`* 요소를 추가할 수 없습니다.

v2

v2에서 `IamPrincipal.ALL`은 모든 보안 주체를 나타냅니다.

다른 유형의 보안 주체에 속한 모든 멤버를 나타내려면 `IamPrincipal`을 만들 때 `IamPrincipalType` 클래스를 사용합니다.

- 모든 사용자에 대한 `IamPrincipal.create(IamPrincipalType.AWS,"*")`입니다.
- 모든 서비스에 대한 `IamPrincipal.create(IamPrincipalType.SERVICE,"*")`입니다.
- 모든 웹 공급자에 대한 `IamPrincipal.create(IamPrincipalType.FEDERATED,"*")`입니다.
- 모든 표준 사용자에 대한 `IamPrincipal.create(IamPrincipalType.CANONICAL_USER,"*")`입니다.

다음 문과 같이 정책 문을 만들 때 `addNotPrincipal` 메서드를 사용하여 *`NotPrincipal`* 요소를 나타낼 수 있습니다.

```
IamPrincipal principal = IamPrincipal.create(IamPrincipalType.AWS,
 "arn:aws:iam::444455556666:root");
IamStatement.builder().addNotPrincipal(principal);
```

AWS SDK for Java의 버전 1에서 버전 2로 DynamoDB 작업 시 변경 사항

주제

AWS SDK for Java의 버전 1과 버전 2 간의 DynamoDB 매핑 API 차이점

DynamoDB 매핑 API는 AWS SDK for Java의 버전 1과 버전 2 간에 크게 변경되었습니다. 버전 1에서는 `DynamoDBMapper`를 사용하여 Java POJO를 작업합니다. 버전 2에서는 업데이트된 메서드 이름, 향상된 스키마 정의 옵션 및 향상된 유형 안전과 함께 `DynamoDbEnhancedClient`를 사용합니다.

주요 차이점은 다음과 같습니다.

- 새 메서드 이름(예: `load` 대신 `getItem`)
- 명시적 테이블 스키마 만들기
- 동기식 및 비동기식 작업 모두에 대한 기본 제공 지원
- 비어 있는 문자열 및 구성 처리 방법 변경

이 섹션에서는 v1 `DynamoDBMapper`에서 v2 `DynamoDbEnhancedClient`로 전환하는 데 도움이 되는 매핑 API 변경 사항, 주석 차이, 구성 업데이트 및 마이그레이션 안내를 다룹니다.

목차

SDK for Java의 버전 1에서 버전 2로의 매핑 라이브러리에 대한 개괄적인 변경 사항

각 라이브러리의 매핑 클라이언트 이름은 V1과 V2에서 다릅니다.

- V1 - DynamoDBMapper
- V2 - DynamoDB 향상된 클라이언트

두 라이브러리와 거의 동일한 방식으로 상호 작용합니다. 즉, 매퍼/클라이언트를 인스턴스화한 다음 이러한 항목을 읽고 DynamoDB 테이블에 쓰는 API에 Java POJO를 제공합니다. 또한 두 라이브러리 모두 POJO 클래스에 대한 주석을 제공하여 클라이언트가 POJO를 처리하는 방법을 지시합니다.

V2로 이전할 때 눈에 띄는 차이점은 다음과 같습니다.

- V2와 V1은 하위 수준 DynamoDB 작업에 서로 다른 메서드 이름을 사용합니다. 예:

V1	V2
로드	getItem
저장	putItem
batchLoad	batchGetItem

- V2는 테이블 스키마를 정의하고 POJO 테이블에 매핑하는 다양한 방법을 제공합니다. 빌더를 사용하여 코드에서 생성된 스키마 또는 주석 사용 중에서 선택할 수 있습니다. 또한 V2는 변경 및 변경 불가능한 버전의 스키마를 제공합니다.
- V2에서는 특히 첫 번째 단계 중 하나로 테이블 스키마를 만드는 반면, V1에서는 필요에 따라 주석이 달린 클래스에서 테이블 스키마를 추론합니다.

- V2는 향상된 클라이언트 API에 문서 API 클라이언트를 포함하는 반면, V1은 별도의 API를 사용합니다.
- 모든 API는 V2의 동기식 및 비동기식 버전에서 사용할 수 있습니다.

V2 향상된 클라이언트에 대한 자세한 내용은 이 가이드의 DynamoDB 매핑 섹션을 참조하세요.

종속성 차이 가져오기

V1

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.amazonaws</gro
upId>
      <artifactId>aws-java-sdk-bom</
artifactId>
      <version> 1.X.X</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>

<dependencies>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-java-sdk-dy
namodb</artifactId>
  </dependency>
</dependencies>
```

V2

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.aw
ssdk</groupId>
      <artifactId>bom</artifactId>
      <version> 2.X.X* </version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>

<dependencies>
  <dependency>
    <groupId>software.amazon.aw
ssdk</groupId>
    <artifactId>dynamodb-enhanced</
artifactId>
  </dependency>
</dependencies>
```

* 최신 버전.

V1에서는 단일 종속성에 하위 수준 DynamoDB API와 매핑/문서 API가 모두 포함되는 반면, V2에서는 dynamodb-enhanced 아티팩트 종속성을 사용하여 매핑/문서 API에 액세스합니다. dynamodb-enhanced 모듈에는 하위 수준 dynamodb 모듈에 대한 전이적 종속성이 포함되어 있습니다.

SDK for Java 버전 1과 버전 2 간의 DynamoDB 매핑 API 변경 사항

클라이언트 만들기

사용 사례	V1	V2
일반 인스턴스화	`AmazonDynamoDB standardClient = AmazonDynamoDBClientBuilder.standard() .withCredentials(credentialsProvider) .withRegion(Regions.US_EAST_1) .build(); DynamoDBMapper mapper = new DynamoDBMapper(standardClient);`	`DynamoDbClient standardClient = DynamoDbClient.builder() .credentialsProvider(ProfileCredentialsProvider.create()) .region(Region.US_EAST_1) .build(); DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder() .dynamoDbClient(standardClient) .build();`
최소 인스턴스화	`AmazonDynamoDB standardClient = AmazonDynamoDBClientBuilder.standard(); DynamoDBMapper mapper = new DynamoDBMapper(standardClient);`	`DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.create();`
속성 변환기 사용*	`DynamoDBMapper mapper = new DynamoDBMapper(standardClient, attributeTransformerInstance);`	`DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder() .dynamoDbClient(standardClient) .extensions(extensionAInstance, extensionBInstance)`

사용 사례	V1	V2
		`.build();`

*V2의 확장은 대략 V1의 속성 변환기에 해당합니다. 이 the section called “확장 프로그램을 사용하여 작업 사용자 지정” 섹션에는 V2의 확장에 대한 자세한 내용이 포함되어 있습니다.

DynamoDB 테이블/인덱스에 대한 매핑 설정

V1에서는 bean 주석을 통해 DynamoDB 테이블 이름을 지정합니다. V2에서는 팩토리 메서드인 `table()`이 원격 DynamoDB 테이블을 나타내는 `DynamoDbTable`의 인스턴스를 생성합니다. `table()` 메서드의 첫 번째 파라미터는 DynamoDB 테이블 이름입니다.

사용 사례	V1	V2

Java POJO 클래스를 DynamoDB 테이블에 매핑	`@DynamoDBTable(tableName ="Customer")`
	`public class Customer {`
	`  ...`
	`}`
		`DynamoDbTable<Customer> customerTable = enhancedClient.table("Customer", TableSchema.fromBean(Customer.class));`

DynamoDB 보조 인덱스에 매핑	1. 인덱스를 나타내는 POJO 클래스를 정의합니다.
	• 인덱스가 있는 테이블의 이름을 제공하는 `@DynamoDBTable` 로 클래스에 주석을 추가합니다.
	• `@DynamoDBIndexHashKey` 및 `@DynamoDBIndexRangeKey` (선택 사항)를 사용하여 속성에 주석을 추가합니다.
	2. 쿼리 표현식을 만듭니다.
		1. `@DynamoDbSecondaryPartitionKey` (GSI의 경우) 및 `@DynamoDbSecondarySortKey` (GSI 또는 LSI의 경우)를 사용하여 POJO 클래스의 속성에 주석을 추가합니다. 예:
		`@DynamoDbSecondarySortKey(indexNames = "IdEmailIndex")`
		`public String getEmail() { return this.email; }`

사용 사례	V1	V2
	3. 인덱스를 나타내는 POJO 클래스에 대한 참조를 사용하여 쿼리합니다. 예 `mapper.query(IdEmailIndex.class, queryExpression)` 여기서 `IdEmailIndex` 는 인덱스의 매핑 클래스입니다. V1 query 메서드에 대해 설명하는 DynamoDB 개발자 안내서의 섹션에는 전체 예제가 나와 있습니다.	2. 인덱스에 대한 참조를 검색합니다. 예: `DynamoDbIndex<Customer> customerIndex = customerTable.index("IdEmailIndex");` 3. 인덱스를 쿼리합니다. 이 안내서의 ??? 섹션에서는 자세한 내용을 제공합니다.

테이블 작업

이 섹션에서는 작업 API를 설명하며, 이는 대부분의 표준 사용 사례에 대한 V1과 V2 간의 차이점이 포함됩니다.

V2에서는 단일 테이블과 관련된 모든 작업이 향상된 클라이언트가 아닌 `DynamoDbTable` 인스턴스에서 호출됩니다. 향상된 클라이언트에는 여러 테이블을 대상으로 지정할 수 있는 메서드가 포함되어 있습니다.

아래 테이블 작업이라는 테이블에서 POJO 인스턴스를 `item` 또는 `customer1` 등의 특정 유형이라고 합니다. V2 예제의 경우 이름이 `table`인 인스턴스는 `DynamoDbTable` 인스턴스에 대한 참조를 반환하는 `enhancedClient.table()`을 이전에 호출한 결과입니다.

대부분의 V2 작업은 표시되지 않은 경우에도 유용한 소비자 패턴으로 호출할 수 있습니다. 예:

```
Customer customer = table.getItem(r # r.key(key));
  or
Customer customer = table.getItem(r # r.key(k ->
 k.partitionValue("id").sortValue("email")))
```

V1 작업의 경우 테이블 작업(아래)에는 모든 오버로드된 양식이 아닌 일반적으로 사용되는 양식 중 일부가 포함됩니다. 예를 들어 `load()` 메서드에는 다음과 같은 오버로드가 있습니다.

```
mapper.load(Customer.class, hashKey)
mapper.load(Customer.class, hashKey, rangeKey)
mapper.load(Customer.class, hashKey, config)
mapper.load(Customer.class, hashKey, rangeKey, config)
mapper.load(item)
mapper.load(item, config)
```

테이블 작업(아래)은 일반적으로 사용되는 양식을 보여줍니다.

```
mapper.load(item)
mapper.load(item, config)
```

테이블 작업

사용 사례	V1	V2

Dynam 테이블에 Java POJO 쓰기
 	
Dynam 작업: PutIte Update m
	`mapper.save(item)`
	`mapper.save(item, config)`
	`mapper.save(item, saveExpression, config)`
	
	V1에서 `DynamoDBMapperConfig.SaveBehavior` 및 주석은 호출할 하위 수준 DynamoDB 메서드를 결정합니다. 일반적으로 `UpdateItem` 은 `SaveBehavior.CLOBBER` 및 `SaveBehavior.PUT` 코드를 사용할 때를 제외하고 호출됩니다. 자동 생성된 키는 특별한 사용 사례이며 경우에 따라 `PutItem` 및 `UpdateItem` 이 모두 사용됩니다.
		`table.putItem(putItemRequest)`
		`table.putItem(item)`
		`table.putItemWithResponse(item) //`
		`Returns metadata.`
		
		`updateItem(updateItemRequest)`
		`table.updateItem(item)`
		`table.updateItemWithResponse(item) //`
		`Returns metadata.`

Dynam 테이	`mapper.load(item)`	`table.getItem(getItemRequest)` `table.getItem(item)`

사용 사례	V1	V2

블에서 Java POJO로의 항목 읽기
 	
Dynam 작업: GetIte
	`mapper.load(item, config)`	`table.getItem(key)`
		`table.getItemWithResponse(key) //`
		`Returns POJO with metadata.`

Dynam 테이블에서 항목 삭제
 	
Dynam 작업: Delete m
	`mapper.delete(item,`
	` deleteExpression, config)`
		`table.deleteItem(deleteItemRequest)`
		`table.deleteItem(item)`
		`table.deleteItem(key)`

사용 사례	V1	V2
Dynam 테이블 또는 보조 인덱스 쿼리 및 페이지 매김 목록 반환 Dynam 작업: Query	`mapper.query(Customer.class , queryExpression) mapper.query(Custom er.class, queryExpression, mapperConfig)`	`table.query(queryRequest) table.query(queryConditional)` 동기식 응답에서 반환된 `PageIterable.strea m()` (지연 로딩) 및 비동기식 응답에서 반환된 `PagePublisher.subscribe()` 사용
Dynam 테이블 또는 보조 인덱스 쿼리 및 목록 반환 Dynam 작업: Query	`mapper.queryPage(Customer.c lass, queryExpression) mapper.queryPage(Cu stomer.class, queryExpr ession, mapperConfig)`	`table.query(queryRequest) table.query(queryConditional)` 동기식 응답에서 반환된 `PageIterable.items ()` (지연 로딩) 및 비동기식 응답에서 반환된 `PagePublisher.items.subscribe()` 사용

사용 사례	V1	V2

Dynam 테이블 또는 보조 인덱스 스캔 및 페이지 매김 목록 반환
 	
Dynam 작업: Scan
	`mapper.scan(Customer.class, scanExpression)`
	`mapper.scan(Customer.class, scanExpression, mapperConfig)`
		`table.scan()`
		`table.scan(scanRequest)`
		
		동기식 응답에서 반환된 `PageIterable.stream()` (지연 로딩) 및 비동기식 응답에서 반환된 `PagePublisher.subscribe()` 사용

Dynam 테이블 또는 보조 인덱스 스캔 및 목록 반환
 	
Dynam 작업: Scan
	`mapper.scanPage(Customer.class, scanExpression)`
	`mapper.scanPage(Customer.class, scanExpression, mapperConfig)`
		`table.scan()`
		`table.scan(scanRequest)`
		
		동기식 응답에서 반환된 `PageIterable.items()` (지연 로딩) 및 비동기식 응답에서 반환된 `PagePublisher.items.subscribe()` 사용

사용 사례	V1	V2

배치의 여러 테이블에서 여러 항목 읽기
 	
DynamoDB 작업: BatchGetItem
	`mapper.batchLoad(Arrays.asList(customer1,`
	
	`  customer2,`
	
	`  book1))`
	`mapper.batchLoad(itemsToGet)`
	`  // itemsToGet: Map<Class<?>, List<KeyPair>>`
		`enhancedClient.batchGetItem(batchGetItemRequest)`
		
		`enhancedClient.batchGetItem(r -> r.readBatches(`
		`  ReadBatch.builder(Record1.class)`
		`  .mappedTableResource(mappedTable1)`
		`  .addGetItem(i -> i.key(k -> k.partitionValue(0)))`
		`  .build(),`
		`  ReadBatch.builder(Record2.class)`
		`  .mappedTableResource(mappedTable2)`
		`  .addGetItem(i -> i.key(k -> k.partitionValue(0)))`
		`  .build()))`
		
		`// Iterate over pages with lazy loading or over all items`
		`  from the same table.`

배치의 여러 테이블에 여러 항목 쓰기
 	
DynamoDB 작업: BatchWriteItem
	`mapper.batchSave(Arrays.asList(customer1,`
	
	`  customer2,`
	
	`  book1))`
		`enhancedClient.batchWriteItem(batchWriteItemRequest)`
		
		`enhancedClient.batchWriteItem(r -> r.writeBatches(`
		`  WriteBatch.builder(Record1.class)`
		`  .mappedTableResource(mappedTable1)`
		`  .addPutItem(item1)`
		`  .build(),`
		`  WriteBatch.builder(Record2.class)`
		`  .mappedTableResource(mappedTable2)`
		`  .addPutItem(item2)`
		`  .build()))`

사용 사례	V1	V2

배치의 여러 테이블에서 여러 항목 삭제
 	
DynamoDB 작업: BatchWriteItem
	`mapper.batchDelete(Arrays.asList(customer1,`
	
	`customer2,`
	
	`book1))`
		`enhancedClient.batchWriteItem(r -> r.writeBatches(`
		`WriteBatch.builder(Record1.class)`
		`.mappedTableResource(mappedTable1)`
		`.addDeleteItem(item1key)`
		`.build(),`
		`WriteBatch.builder(Record2.class)`
		`.mappedTableResource(mappedTable2)`
		`.addDeleteItem(item2key)`
		`.build()))`

배치에서 여러 항목 쓰기/삭제
 	
DynamoDB 작업: BatchWriteItem
	`mapper.batchWrite(Arrays.asList(customer1, book1),`
	`Arrays.asList(customer2))`
		`enhancedClient.batchWriteItem(r -> r.writeBatches(`
		`WriteBatch.builder(Record1.class)`
		`.mappedTableResource(mappedTable1)`
		`.addPutItem(item1)`
		`.build(),`
		`WriteBatch.builder(Record2.class)`
		`.mappedTableResource(mappedTable2)`
		`.addDeleteItem(item2key)`
		`.build()))`

사용 사례	V1	V2
트랜잭션 쓰기 수행 DynamoDB 작업: TransactWriteItems	`mapper.transactionWrite(transactionWriteRequest)`	`enhancedClient.transactWriteItems(transasctWriteItemsRequest)`
트랜잭션 읽기 수행 DynamoDB 작업: TransactGetItems	`mapper.transactionLoad(transactionLoadRequest)`	`enhancedClient.transactGetItems(transactGetItemsRequest)`

사용 사례	V1	V2

쿼리의 일치하는 항목 수 가져오기
 	
DynamoDB 작업: Select COUNT 로 Query
	`mapper.count(Customer.class, queryExpression)`	`// Get the count from query results.`
		`PageIterable<Customer> pageIterable = customerTable.query(QueryEnhancedRequest.builder()`
		`.queryConditional(queryConditional)`
		`.select(Select.COUNT)`
		`.build());`
		`Iterator<Page<Customer>> iterator = pageIterable.iterator();`
		`Page<Customer> page = iterator.next();`
		`int count = page.count();`
		
		`// For a more concise approach, you can chain the method calls:`
		`int count = customerTable.query(QueryEnhancedRequest.builder()`
		`.queryConditional(queryConditional)`
		`.select(Select.COUNT)`
		`.build())`
		`.iterator().next().count();`

사용 사례	V1	V2

스캔의 일치하는 항목 수 가져오기
 	
DynamoDB 작업: Select UNT 로 Scan
	`mapper.count(Customer.class , scanExpression)`	`// Get the count from scan results.`
		`PageIterable<Customer> pageIterable =`
		`customerTable.scan(ScanEnhancedRequest.builder()`
		`.filterExpression(filterExpression)`
		`.select(Select.COUNT)`
		`.build());`
		`Iterator<Page<Customer>> iterator = pageIterable.iterator();`
		`Page<Customer> page = iterator.next();`
		`int count = page.count();`
		
		`// For a more concise approach, you can chain the method calls:`
		`int count = customerTable.scan(ScanEnhancedRequest.builder()`
		`.filterExpression(filterExpression)`
		`.select(Select.COUNT)`
		`.build())`
		`.iterator().next().count();`

<table>
<tr><th>사용 사례</th><th>V1</th><th>V2</th></tr>
<tr><td>Dynam에서 POJO 클래스에 해당하는 테이블 만들기

Dynam 작업: Create le</td><td><pre>mapper.generateCreateTableRequest(Customer.class)</pre>이전 문은 하위 수준 테이블 만들기 요청을 생성합니다. 사용자는 DynamoDB 클라이언트에서 <code>createTable</code> 을 호출해야 합니다.</td><td><pre>table.createTable(createTableRequest)

table.createTable(r -> r.provisionedThroughput(defaultThroughput())
    .globalSecondaryIndices(
        EnhancedGlobalSecondaryIndex.builder()
            .indexName("gsi_1")
            .projection(p -> p.projectionType(ProjectionType.ALL))
            .provisionedThroughput(defaultThroughput())
            .build()));</pre></td></tr>
<tr><td>Dynam에서 병렬 스캔 수행

Dynam 작업: Segmer 및 Totals ents 파라미터로 Scan</td><td><pre>mapper.parallelScan(Customer.class,
                    scanExpression,
                    numTotalSegments)</pre></td><td>사용자는 작업자 스레드를 처리하고 각 세그먼트에 대해 <code>scan</code>을 호출해야 합니다.<pre>table.scan(r -> r.segment(0).totalSegments(5))</pre></td></tr>
</table>

사용 사례	V1	V2
Amazon S3를 DynamoDB와 통합하여 지능형 S3 링크 저장	`mapper.createS3Link(bucket, key)` `mapper.getS3ClientCache()`	Amazon S3와 DynamoDB를 통합하므로 지원되지 않습니다.

클래스 및 속성 매핑

V1과 V2 모두에서 bean 스타일 주석을 사용하여 클래스를 테이블에 매핑합니다. V2는 변경 불가능한 클래스 작업 등 특정 사용 사례에 대한 스키마를 정의하는 다른 방법도 제공합니다.

bean 주석

다음 테이블에는 V1 및 V2에 사용되는 특정 사용 사례별로 동등한 bean 주석이 나와 있습니다. `Customer` 클래스 시나리오는 파라미터를 설명하는 데 사용됩니다.

V2의 주석, 클래스, 열거는 Camel 대문자 규칙을 따르고 'DynamoDB'가 아닌 'DynamoDb'를 사용합니다.

사용 사례	V1	V2

클래스를 테이블에 매핑	`@DynamoDBTable (tableName ="CustomerTable")`	`@DynamoDbBean`
		`@DynamoDbBean(converterProviders = {...})`
		테이블 이름은 `DynamoDbEnhancedClient#table()` 메서드를 호출할 때 정의됩니다.

사용 사례	V1	V2
클래스 멤버를 테이블 속성으로 지정	`@DynamoDBAttribute (attributeName = "customerName")`	`@DynamoDbAttribute("customerName")`
클래스 멤버를 해시/파티션 키로 지정	`@DynamoDBHashKey`	`@DynamoDbPartitionKey`
클래스 멤버를 범위/정렬 키로 지정	`@DynamoDBRangeKey`	`@DynamoDbSortKey`
클래스 멤버를 보조 인덱스 해시/파티션 키로 지정	`@DynamoDBIndexHashKey`	`@DynamoDbSecondaryPartitionKey`
클래스 멤버를 보조 인덱스 범위/정렬 키로 지정	`@DynamoDBIndexRangeKey`	`@DynamoDbSecondarySortKey`
테이블에 매핑할 때 이 클래스 멤버 무시	`@DynamoDBIgnore`	`@DynamoDbIgnore`
클래스 멤버를 자동 생성된 UUID 키 속성으로 지정	`@DynamoDBAutoGeneratedKey`	`@DynamoDbAutoGeneratedUuid` 이를 제공하는 확장 프로그램은 기본적으로 로드되지 않으므로 클라이언트 빌더에 확장 프로그램을 추가해야 합니다.

사용 사례	V1	V2
클래스 멤버를 자동 생성된 타임스탬프 속성으로 지정	`@DynamoDBAutoGeneratedTimestamp`	`@DynamoDbAutoGeneratedTimestampAttribute` 이를 제공하는 확장 프로그램은 기본적으로 로드되지 않으므로 클라이언트 빌더에 확장 프로그램을 추가해야 합니다.
클래스 멤버를 자동 증분 버전 속성으로 지정	`@DynamoDBVersionAttribute`	`@DynamoDbVersionAttribute` 이를 제공하는 확장 프로그램은 자동으로 로드됩니다.
클래스 멤버를 사용자 지정 변환이 필요한 것으로 지정	`@DynamoDBTypeConverted`	`@DynamoDbConvertedBy`
다른 속성 유형으로 저장할 클래스 멤버 지정	`@DynamoDBTyped(<DynamoDBAttributeType>)`	`AttributeConverter` 구현을 사용합니다. V2는 일반적인 Java 유형을 위한 많은 기본 제공 변환기를 지원합니다. 자체 사용자 지정 `AttributeConverter` 또는 `AttributeConverterProvider` 를 구현할 수도 있습니다. 이 설명서의 the section called “제어 속성 변환” 섹션을 참조하세요.
DynamoDB 문서(JSON 형식 문서) 또는 하위 문서로 직렬화할 수 있는 클래스 지정	`@DynamoDBDocument`	향상된 문서 API 사용 다음 리소스를 참조하세요. • the section called “JSON 문서로 작업”이 가이드의 • 향상된 문서 API가 릴리스될 때 게시된 블로그 게시물

V2 추가 주석

사용 사례	V1	V2
Java 값이 null인 경우 클래스 멤버를 null 속성으로 저장하지 않도록 지정	N/A	`@DynamoDbIgnoreNulls`
모든 속성이 null인 경우 클래스 멤버를 비어 있는 객체로 지정	N/A	`@DynamoDbPreserveEmptyObject`
클래스 멤버에 대한 특별 업데이트 작업 지정	N/A	`@DynamoDbUpdateBehavior`
변경할 수 없는 클래스 지정	N/A	`@DynamoDbImmutable`
클래스 멤버를 자동 증분 카운터 속성으로 지정	N/A	`@DynamoDbAtomicCounter` 이 기능을 제공하는 확장 프로그램은 자동으로 로드됩니다.

구성

V1에서는 일반적으로 `DynamoDBMapperConfig`의 인스턴스를 사용하여 특정 동작을 제어합니다. 매퍼를 만들거나 요청할 때 구성 객체를 제공할 수 있습니다. V2에서 구성은 작업에 대한 요청 객체에 따라 다릅니다.

사용 사례	V1	V1의 기본값	V2
	`DynamoDBMapperConfig.builder()`		

사용 사례	V1	V1의 기본값	V2
배치 로드/쓰기 재시도 전략	`.withBatchLoadRetryStrategy(loadRetryStrategy)` `.withBatchWriteRetryStrategy(writeRetryStrategy)`	실패한 항목 재시도	기본 DynamoDBClient 에서 재시도 전략을 구성합니다. 이 설명서의 the section called “Retries” 섹션을 참조하세요.
일관된 읽기	`.withConsistentReads(CONSISTENT)`	EVENTU	기본적으로 일관된 읽기는 읽기 작업에 대해 false입니다. 요청 객체에서 `.consistentRead(true)` 로 재정의합니다.
마샬러/언마샬러 세트가 포함된 변환 스키마	`.withConversionSchema(conversionSchema)` 정적 구현에서는 구버전과 이전 버전과의 호환성을 제공합니다.	V2_COMPATIBLE	해당 사항 없음. 이는 DynamoDB(V1)의 최신 버전이 데이터 형식을 저장한 방식을 나타내는 레거시 기능이며, 이 동작은 향상된 클라이언트에서 유지되지 않습니다. DynamoDB V1의 동작 예제는 부울을 부울 대신 숫자로 저장합니다.

테이블 이름	`.withObjectTableNameResolver()`
	`.withTableNameOverride()`
	`.withTableNameResolver()`
	정적 구현에서 구버전과 이전 버전과의 호환성 제공
		클래스에서 주석 또는 추측 사용	테이블 이름은 DynamoDbEnhancedClient#table() 메서드를 호출할 때 정의됩니다.

사용 사례	V1	V1의 기본값	V2
페이지 매김 로드 전략	`.withPaginationLoadingStrategy(strategy)` 옵션은 LAZY_LOADING, EAGER_LOADING 또는 ITERATION_ONLY 입니다.	LAZY_LOADING	• 반복만 기본값입니다. 다른 V1 옵션은 지원되지 않습니다. • 다음을 사용하여 V2에서 동등한 즉시 로딩을 구현할 수 있습니다. `List<Customer> allItems = customerTable.scan().items().stream().collect(Collectors.toList());` • 지연 로딩의 경우 액세스한 항목에 대한 자체 캐싱 로직을 구현해야 합니다.
지표 수집 요청	`.withRequestMetricCollector(collector)`	null	표준 DynamoDB 클라이언트를 구축할 때 ClientOverrideConfiguration 에서 metricPublisher() 를 사용합니다.
동작 저장	`.withSaveBehavior(SaveBehavior.CLOBBER)` 옵션은 UPDATE, CLOBBER, PUT, APPEND_SET , UPDATE_SKIP_NULL_ATTRIBUTES 입니다.	UPDATE	V2에서는 putItem() 또는 updateItem() 을 명시적으로 호출합니다. CLOBBER or PUT: v 2의 해당 작업은 putItem() 을 호출합니다. 특정 CLOBBER 구성은 없습니다. UPDATE에 해당합니다.updateItem() UPDATE_SKIP_NULL_ATTRIBUTES 에 해당합니다.updateItem() 요청 설정 ignoreNulls 및 주석/태그 DynamoDbUpdateBehavior 를 사용하여 업데이트 동작을 제어합니다. APPEND_SET : 지원되지 않음:

사용 사례	V1	V1의 기본값	V2
유형 변환기 팩토리	`.withTypeConverterFactory(typeConverterFactory)`	표준 유형 변환기	bean에 설정하는 수단 `@DynamoDbBean(converterProviders = {ConverterProvider.class, DefaultAttributeConverterProvider.class})`

작업별 구성

V1에서 `query()` 등의 일부 작업은 작업에 제출된 '표현식' 객체를 통해 고도로 구성할 수 있습니다. 예:

```
DynamoDBQueryExpression<Customer> emailBwQueryExpr = new
 DynamoDBQueryExpression<Customer>()
    .withRangeKeyCondition("Email",
        new Condition()
            .withComparisonOperator(ComparisonOperator.BEGINS_WITH)
            .withAttributeValueList(
                new AttributeValue().withS("my")));

mapper.query(Customer.class, emailBwQueryExpr);
```

V2에서는 구성 객체를 사용하는 대신 빌더를 사용하여 요청 객체에 파라미터를 설정합니다. 예:

```
QueryEnhancedRequest emailBw = QueryEnhancedRequest.builder()
    .queryConditional(QueryConditional
        .sortBeginsWith(kb -> kb
            .sortValue("my"))).build();

customerTable.query(emailBw);
```

조건

V2에서 조건부 및 필터링 표현식은 조건과 이름 및 필터 매핑을 캡슐화하는 `Expression` 객체를 사용하여 표현됩니다.

사용 사례	작업	V1	V2
예상 속성 조건	save(), delete(), query(), scan()	`new DynamoDBSaveExpression()  .withExpected(Collections.singletonMap(  "otherAttribute", new ExpectedAttributeValue(false)))  .withConditionalOperator(ConditionalOperator.AND);`	지원 종료되며 대신 `ConditionExpression` 을 사용합니다.
조건 표현식	delete()	`deleteExpression.setConditionExpression("zipcode = :zipcode") deleteExpression.setExpressionAttributeValues(...)`	`Expression conditionExpression = Expression.builder() .expression("#key = :value OR #key1 = :value1") .putExpressionName("#key", "attribute") .putExpressionName("#key1", "attribute3") .putExpressionValue(":value", AttributeValues.stringValue("wrong")) .putExpressionValue(":value1", AttributeValues.stringValue("three")) .build(); DeleteItemEnhancedRequest request = DeleteItemEnhancedRequest.builder() .conditionExpression(conditionExpression).build();`

사용 사례	작업	V1	V2
필터 표현식	query(), scan()	`scanExpression .withFilterExpression("#statename = :state") .withExpressionAttributeValues(attributeValueMapBuilder.build()) .withExpressionAttributeNames(attributeNameMapBuilder.build())`	`Map<String, AttributeValue> values = singletonMap(":key", stringValue("value")); Expression filterExpression = Expression.builder() .expression("name = :key") .expressionValues(values) .build(); QueryEnhancedRequest request = QueryEnhancedRequest.builder() .filterExpression(filterExpression).build();`
쿼리에 대한 키 조건 표현식	query()	`queryExpression.withKeyConditionExpression()`	`QueryConditional keyEqual = QueryConditional.keyEqualTo(b -> b .partitionValue("movie01")); QueryEnhancedRequest tableQuery = QueryEnhancedRequest.builder() .queryConditional(keyEqual) .build();`

유형 변환

기본 변환기

V2에서 SDK는 모든 일반 유형에 대한 기본 변환기 세트를 제공합니다. 전반적 공급자 수준과 단일 속성 모두에서 유형 변환기를 변경할 수 있습니다. AttributeConverter API 참조에서 사용 가능한 변환기 목록을 확인할 수 있습니다.

속성에 대한 사용자 지정 변환기 설정

V1에서는 @DynamoDBTypeConverted로 getter 메서드에 주석을 달아 Java 속성 유형과 DynamoDB 속성 유형 간에 변환되는 클래스를 지정할 수 있습니다. 예를 들어 Java Currency 유형과 DynamoDB 문자열 간에 변환되는 CurrencyFormatConverter는 다음 코드 조각과 같이 적용할 수 있습니다.

```
@DynamoDBTypeConverted(converter = CurrencyFormatConverter.class)
public Currency getCurrency() { return currency; }
```

이전 코드 조각과 동일한 V2가 아래에 나와 있습니다.

```
@DynamoDbConvertedBy(CurrencyFormatConverter.class)
public Currency getCurrency() { return currency; }
```

> **Note**
>
> V1에서는 속성 자체, 유형 또는 사용자 정의 주석에 주석을 적용할 수 있으며, V2는 주석을 getter에만 적용할 수 있도록 지원합니다.

유형 변환기 팩토리 또는 공급자 추가

V1에서는 고유한 유형 변환기 세트를 제공하거나 구성에 유형 변환기 팩토리를 추가하여 관심 있는 유형을 재정의할 수 있습니다. 유형 변환기 팩토리는 DynamoDBTypeConverterFactory를 확장하며, 기본 세트에 대한 참조를 가져와서 확장하면 재정의가 수행됩니다. 다음은 이 작업을 수행하는 방법을 보여주는 Java 코드 조각입니다.

```
DynamoDBTypeConverterFactory typeConverterFactory =
    DynamoDBTypeConverterFactory.standard().override()
        .with(String.class, CustomBoolean.class, new DynamoDBTypeConverter<String,
 CustomBoolean>() {
            @Override
            public String convert(CustomBoolean bool) {
                return String.valueOf(bool.getValue());
            }
            @Override
            public CustomBoolean unconvert(String string) {
                return new CustomBoolean(Boolean.valueOf(string));
            }}).build();
```

```
DynamoDBMapperConfig config =
    DynamoDBMapperConfig.builder()
        .withTypeConverterFactory(typeConverterFactory)
        .build();
DynamoDBMapper mapperWithTypeConverterFactory = new DynamoDBMapper(dynamo, config);
```

V2는 @DynamoDbBean 주석을 통해 유사한 기능을 제공합니다. 단일 AttributeConverterProvider 또는 순서대로 정리된 AttributeConverterProvider의 체인을 제공할 수 있습니다. 자체 속성 변환기 공급자 체인을 제공하는 경우 기본 변환기 공급자가 재정의되고 해당 속성 변환기를 사용하려면 체인에 포함되어야 함을 참고하시기 바랍니다.

```
@DynamoDbBean(converterProviders = {
   ConverterProvider1.class,
   ConverterProvider2.class,
   DefaultAttributeConverterProvider.class})
public class Customer {
  ...
}
```

이 안내서의 속성 변환 섹션에는 V2에 대한 전체 예제가 포함되어 있습니다.

SDK for Java 버전 1과 버전 2 간의 문자열 처리 차이점

V1과 V2는 DynamoDB로 데이터를 전송할 때 비어 있는 문자열을 다르게 처리합니다.

- V1: DynamoDB로 전송하기 전에 비어 있는 문자열을 null 값으로 변환합니다(속성 없음).
- V2: 비어 있는 문자열을 DynamoDB에 실제 비어 있는 문자열 값으로 전송합니다.

> **Important**
>
> V2로 마이그레이션한 후 DynamoDB에 비어 있는 문자열을 저장하지 않으려면 사용자 지정 변환기를 구현해야 합니다. 사용자 지정 변환기가 없으면 V2는 비어 있는 문자열을 DynamoDB 항목에 실제 비어 있는 문자열 속성으로 저장합니다. 이는 이러한 속성을 완전히 생략하는 V1의 동작과 다릅니다.

Example비어 있는 문자열 속성을 null로 변환하는 V2용 사용자 지정 변환기

```
/**
```

```
 * Custom converter that maintains V1 behavior by converting empty strings to null
 values
 * when writing to DynamoDB, ensuring compatibility with existing data. No attribute
 will be saved to DynamoDB.
 */
public class NullifyEmptyStringConverter implements AttributeConverter<String> {
    @Override
    public AttributeValue transformFrom(String value) {
        if (value == null || value.isEmpty()) {
            return AttributeValue.builder().nul(true).build();
        }
        return AttributeValue.builder().s(value).build();
    }

    @Override
    public String transformTo(AttributeValue attributeValue) {
        if (attributeValue.nul()) {
            return null;
        }
        return attributeValue.s();
    }

    @Override
    public EnhancedType<String> type() {
        return EnhancedType.of(String.class);
    }

    @Override
    public AttributeValueType attributeValueType() {
        return AttributeValueType.S;
    }
}

// V2 usage:
@DynamoDbBean
public class Customer {
    private String name;

    @DynamoDbConvertedBy(NullifyEmptyStringConverter.class)
    public String getName() {
        return name;
    }
}
```

SDK for Java 버전 1과 버전 2의 낙관적 잠금 차이점

V1과 V2 모두 bean 클래스에 하나의 속성을 표시하여 버전 번호를 저장하는 속성 주석으로 낙관적 잠금을 구현합니다.

낙관적 잠금 동작의 차이점

	V1	V2
bean 클래스 주석	`@DynamoDBVersionAttribute`	`@DynamoDbVersionAttribute` (V2는 소문자 'b' 사용)
초기 저장	버전 번호 속성은 1로 설정됩니다.	`@DynamoDbVersionAttribute(startAt = X)`로 설정된 버전 속성의 시작 값입니다. 기본값은 0입니다.
업데이트	조건부 확인에서 업데이트 중인 객체의 버전 번호가 데이터베이스의 번호와 일치하는지 확인하면 버전 번호 속성이 1씩 증가합니다.	조건부 확인에서 업데이트 중인 객체의 버전 번호가 데이터베이스의 번호와 일치하는지 확인하면 버전 번호 속성이 증분합니다. `@DynamoDbVersionAttribute(incrementBy = X)`로 설정된 `incrementBy` 옵션에 따라 증분하는 버전 번호 속성입니다. 기본 값은 1입니다.
삭제	`DynamoDBMapper` 는 삭제 중인 객체의 버전 번호가 데이터베이스의 버전 번호와 일치하는지 조건부 확인을 추가합니다.	V2는 삭제 작업에 대한 조건을 자동으로 추가하지 않습니다. 삭제 동작을 제어하려면 조건 표현식을 수동으로 추가해야 합니다. 다음 예제에서 `recordVersion` 은 bean의 버전 속성입니다. `// 1. Read the item and get its current version.` `Customer item = customerTable.getItem(Key.builder().` `partitionValue("someId").build());` `AttributeValue currentVersion = item.getRecordVers` `ion();` `// 2. Create conditional delete with the `currentV` `ersion` value.` `DeleteItemEnhancedRequest deleteItemRequest =`

	V1	V2
		DeleteItemEnhancedRequest.builder() .key(KEY) .conditionExpression(Expression.builder() .expression("recordVersion = :current_version_value") .putExpressionValue(":current_version_value", currentVersion) .build()).build(); customerTable.deleteItem(deleteItemRequest);
조건 확인을 사용한 트랜잭션 쓰기	addConditionCheck 메서드에서 @DynamoDBVersionAttribute 로 주석이 달린 bean 클래스는 사용할 수 없습니다.	transactWriteItems 요청에 대해 addConditionCheck 빌더 메서드의 @DynamoDbVersionAttribute 주석과 함께 bean 클래스를 사용할 수 있습니다.
비활성화	낙관적 잠금은 DynamoDBMapperConfig.SaveBehavior 열거 값을 UPDATE에서 CLOBBER로 변경하면 사용 해제할 수 있습니다.	@DynamoDbVersionAttribute 주석을 사용하지 마세요.

SDK for Java 버전 1과 버전 2 간의 유용한 setter 차이점

V1용 DynamoDB 매핑 API에서 유용한 setter와 버전 2.30.29 이후 V2에서 POJO를 사용할 수 있습니다.

예를 들어 다음 POJO는 `setName` 메서드에서 `Customer` 인스턴스를 반환합니다.

```
// V1

@DynamoDBTable(tableName ="Customer")
public class Customer{
  private String name;
  // Other attributes and methods not shown.
  public Customer setName(String name){
     this.name = name;
     return this;
  }
}
```

그러나 2.30.29 이전의 V2 버전을 사용하는 경우 `setName`은 `name` 값이 `null`인 `Customer` 인스턴스를 반환합니다.

```
// V2 prior to version 2.30.29.

@DynamoDbBean
public class Customer{
  private String name;
  // Other attributes and methods not shown.
  public Customer setName(String name){
     this.name = name;
     return this;  // Bug: returns this instance with a `name` value of `null`.
  }
}
```

```
// Available in V2 since version 2.30.29.

@DynamoDbBean
public class Customer{
  private String name;
  // Other attributes and methods not shown.
  public Customer setName(String name){
     this.name = name;
```

```
        return this;  // Returns this instance for method chaining with the `name` value
 set.
  }
}
```

AWS SDK for Java의 버전 1과 버전 2 간의 문서 API 차이점

문서 API는 DynamoDB 테이블에서 JSON 형식 문서를 단일 항목으로 사용할 수 있도록 지원합니다. V1 문서 API는 V2에 해당 API가 있지만, V1처럼 문서 API에 별도의 클라이언트를 사용하는 대신, V2에서는 DynamoDB 향상된 클라이언트에 문서 API 기능을 통합했습니다.

V1에서 `Item` 클래스는 DynamoDB 테이블의 비정형 레코드를 나타냅니다. V2에서 비정형 레코드는 `EnhancedDocument` 클래스의 인스턴스로 표시됩니다. 프라이머리 키는 V2의 테이블 스키마와 V1의 항목 자체에 정의되어 있습니다.

아래 테이블에서는 V1과 V2의 문서 API를 비교합니다.

사용 사례	V1	V2
Create a document client	`AmazonDynamoDB client = ... //Create a client. DynamoDB documentClient = new DynamoDB(client);`	`// The V2 Document API uses the same DynamoDbEnhancedClient // that is used for mapping POJOs. DynamoDbClient standardClient = ... //Create a standard client. DynamoDbEnhancedClient enhancedClient = ... // Create an enhanced client.`
Reference a table	`Table documentTable = docClient.documentClient("Person");`	`DynamoDbTable<EnhancedDocument> documentTable = enhancedClient.table("Person", TableSchema.documentSchemaBuilder()`

사용 사례	V1	V2
		.addIndex PartitionKey(Table Metadata.primaryIn dexName(),"id", AttributeValueType.S) .attribut eConverterProvider s(AttributeConvert erProvider.default Provider()) .build()) ;
Work with semi-structured data		
Put item	Item item = new Item() .withPrimaryKey("i d", 50) .withString("first Name", "Shirley"); PutItemOutcome outcome = documentTable.putI tem(item);	EnhancedDocument personDocument = EnhancedDocument.b uilder() .putNumbe r("id", 50) .putStrin g("firstName", "Shirley") .build(); documentTable.put Item(personDocument);

Get item	GetItemOutcome outcome
	= documentTable.getI
	temOutcome("id", 50);
	Item personDocFromDb =
	outcome.getItem();
	String firstName =
	personDocFromDb.ge
	tString("firstName");
		EnhancedDocument
		personDocFromDb =
		documentTable
		.getItem(
		Key.builder()
		.partitio
		nValue(50)
		.build());
		String firstName =
		personDocFromDb.ge
		tString("firstName");

사용 사례	V1	V2
Work with JSON items		
Convert a JSON structure to use it with the Document API	`// The 'jsonPerson' identifier is a JSON string. Item item = new Item().fromJSON(jsonPerson);`	`// The 'jsonPerson' identifier is a JSON string. EnhancedDocument document = EnhancedDocument.builder() .json(jsonPerson).build());`
Put JSON	`documentTable.putItem(item)`	`documentTable.putItem(document);`
Read JSON	`GetItemOutcome outcome = //Get item. String jsonPerson = outcome.getItem().toJSON();`	`String jsonPerson = documentTable.getItem(Key.builder() .partitionValue(50).build()) .fromJson();`

문서 API에 대한 API 참조 및 안내서

	V1	V2
API 참조	API 참조	API 참조
문서 안내서	Amazon DynamoDB 개발자 안내서	향상된 문서 API(이 안내서)

V1 Xpec API에서 V2 표현식 API

문서 중심 데이터로 작업할 표현식을 만드는 데 도움이 되는 V1에서 사용할 수 있는 표현식 사양(Xspec) API는 V2에서 사용할 수 없습니다. V2는 문서 중심 데이터와 object-to-item 매핑 데이터 모두에서 작동하는 표현식 API를 사용합니다.

	V1	V2
API 이름	표현식 사양(Xspec) API	표현식 API
함께 작동	updateItem 및 scan 등 문서 API 테이블 클래스의 메서드	DynamoDB 향상된 클라이언트의 두 API: 1. object-to-item 매핑 API의 메서드 2. 문서 중심 데이터(JSON) 작업용 향상된 문서 API의 메서드 DynamoDbTable 인스턴스를 획득한 후 두 유형의 API에 대해 다음을 수행합니다. • BeanTableSchema 에서 예를 들어 object-to-item 매핑 API인 경우 • DocumentTableSchema 에서 문서 중심 API인 경우 요청 객체를 만들 때 DynamoDbTable 메서드에서 표현식을 사용합니다. QueryEnhancedRequest.Builder 의 filterExpression 메서드에서의 예제
리소스	• Xpec 초기 릴리스 블로그 게시물 • API 참조	이 Java 개발자 안내서의 표현식 정보

암호화 라이브러리 마이그레이션

DynamoDB가 Java SDK의 V2로 작동하도록 암호화 라이브러리를 마이그레이션하는 방법에 대한 자세한 내용은 Amazon DynamoDB Encryption Client 개발자 안내서를 참조하세요.

버전 1에서 버전 2로 자동 Amazon SQS 요청 배치 처리 변경 사항

이 주제에서는 AWS SDK for Java의 버전 1과 버전 2 간에 Amazon SQS에 대한 자동 요청 배치 처리의 변경 사항을 자세히 설명합니다.

높은 수준의 변경 사항

AWS SDK for Java 1.x는 요청 배치 처리를 위해 명시적으로 초기화해야 하는 별도의 `AmazonSQSBufferedAsyncClient` 클래스를 사용하여 클라이언트 측 버퍼링을 수행합니다.

AWS SDK for Java 2.x는 `SqsAsyncBatchManager`를 사용하여 버퍼링 기능을 간소화하고 개선합니다. 이 인터페이스의 구현은 표준 `SqsAsyncClient`와 직접 통합된 자동 요청 배치 처리 기능을 제공합니다. v2의 `SqsAsyncBatchManager`에 대해 알아보려면 이 안내서의 the section called “자동 요청 배치 처리 사용” 주제를 참조하세요.

변경 사항	v1	v2

Maven 종속성	`<dependencyManagement>`
	`  <dependencies>`
	`  <dependency>`
	`  <groupId>com.amazonaws</groupId>`
	`  <artifactId>aws-java-sdk-bom</artifactId>`
	`  <version>1.12.782[1]</version>`
	`  <type>pom</type>`
	`  <scope>import</scope>`
	`  </dependency>`
	`  </dependencies>`
	`</dependencyManagement>`
		`<dependencyManagement>`
		`  <dependencies>`
		`  <dependency>`
		`  <groupId>software.amazon.awssdk</groupId>`
		`  <artifactId>bom</artifactId>`
		`  <version>2.31.15[2]</version>`
		`  <type>pom</type>`
		`  <scope>import</scope>`
		`  </dependency>`
		`  </dependencies>`
		`</dependencyManagement>`
		`<dependencies>`

변경 사항	v1	v2
	<dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-sqs</ artifactId> </dependency> </dependencies>	<dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>sqs</artifactId> </dependency> </dependencies>
패키지 이름	com.amazonaws.serv ices.sqs.buffered	software.amazon.aw ssdk.services.sqs. batchmanager
클래스 이름	AmazonSQSBufferedA syncClient	SqsAsyncBatchManager

[1] 최신 버전. [2] 최신 버전.

자동 SQS 요청 배치 처리 사용

변경 사항	v1	v2

배치 관리자 만들기	AmazonSQSAsync sqsAsync
	 = new AmazonSQS
	AsyncClient();
	AmazonSQSAsync
	 bufferedSqs = new
	  AmazonSQS
	BufferedAsyncClien
	t(sqsAsync);
		SqsAsyncClient
		 asyncClient =
		 SqsAsyncClient.cre
		ate();
		SqsAsyncBatchManager
		 sqsAsyncBatchManager
		 =
		  asyncClie
		nt.batchManager();

사용자 지정 구성을 사용하여 배치 관리자 만들기	AmazonSQSAsync sqsAsync = new AmazonSQS AsyncClient();	BatchOverrideConfi guration batchOver rideConfiguration =

변경 사항	v1	v2
	QueueBufferConfig queueBufferConfig = new QueueBufferConfig() .withMaxBatchOpenMs(200) .withMaxBatchSize(10) .withMinReceiveWaitTimeMs(1000) .withVisibilityTimeoutSeconds(20) .withReceiveMessageAttributeNames(messageAttributeValues); AmazonSQSAsync bufferedSqs = new AmazonSQSBufferedAsyncClient(sqsAsync, queueBufferConfig);	BatchOverrideConfiguration.builder() .sendRequestFrequency(Duration.ofMillis(200)) .maxBatchSize(10) .receiveMessageMinWaitDuration(Duration.ofMillis(1000)) .receiveMessageVisibilityTimeout(Duration.ofSeconds(20)) .receiveMessageSystemAttributeNames(messageSystemAttributeNames) .receiveMessageAttributeNames(messageAttributeValues) .build(); SqsAsyncBatchManager sqsAsyncBatchManager = SqsAsyncBatchManager.builder() .overrideConfiguration(batchOverrideConfiguration) .client(SqsAsyncClient.create()) .scheduledExecutor(Executors.newScheduledThreadPool(8)) .build();

변경 사항	v1	v2
메시지 전송	`Future<SendMessageResult> sendResultFuture = bufferedSqs.sendMessageAsync(new SendMessageRequest() .withQueueUrl(queueUrl) .withMessageBody(body));`	`CompletableFuture<SendMessageResponse> sendCompletableFuture = sqsAsyncBatchManager.sendMessage( SendMessageRequest.builder() .queueUrl(queueUrl) .messageBody(body) .build());`
메시지 삭제	`Future<DeleteMessageResult> deletResultFuture = bufferedSqs.deleteMessageAsync(new DeleteMessageRequest() .withQueueUrl(queueUrl));`	`CompletableFuture<DeleteMessageResponse> deleteResultCompletableFuture = sqsAsyncBatchManager.deleteMessage( DeleteMessageRequest.builder() .queueUrl(queueUrl) .build());`

변경 사항	v1	v2

메시지의 표시 여부 변경	`Future<ChangeMessageVisibilityResult> changeVisibilityResultFuture = bufferedSqs.changeMessageVisibilityAsync(new ChangeMessageVisibilityRequest()`
	
	`.withQueueUrl(queueUrl)`
	
	`.withVisibilityTimeout(20));`
		`CompletableFuture<ChangeMessageVisibilityResponse> changeResponseCompletableFuture = sqsAsyncBatchManager.changeMessageVisibility(`
		
		`ChangeMessageVisibilityRequest.builder()`
		
		`.queueUrl(queueUrl)`
		
		`.visibilityTimeout(20)`
		
		`.build());`

메시지 수신	`ReceiveMessageResult receiveResult = bufferedSqs.receiveMessage(new ReceiveMessageRequest()`
	
	`.withQueueUrl(queueUrl));`
		`CompletableFuture<ReceiveMessageResponse> responseCompletableFuture = sqsAsyncBatchManager.receiveMessage(`
		
		`ReceiveMessageRequest.builder()`
		
		`.queueUrl(queueUrl)`
		
		`.build());`

비동기식 반환 유형 차이점

변경 사항	v1	v2
반환 타입	`Future<ResultType>`	`CompletableFuture<ResponseType>`
콜백 메커니즘	별도의 `onSuccess` 및 `onError` 메서드가 있는 `AsyncHandler` 필요	`whenComplete()`, `thenCompose()`, `thenApply()` 등 JDK에서 제공하는 `CompletableFuture` API 사용
예외 처리	`AsyncHandler#onError()` 메서드 사용	`exceptionally()`, `handle()`, `whenComplete()` 등 JDK에서 제공하는 `CompletableFuture` API 사용
취소	`Future.cancel()`을 통한 기본 지원	상위 `CompletableFuture`를 취소하면 체인의 모든 종속 future 자동 취소

비동기식 완료 처리 차이점

변경 사항	v1	v2
응답 핸들러 구현	`Future<ReceiveMessageResult> future = bufferedSqs.receiveMessageAsync( receiveRequest, new AsyncHandler<ReceiveMessageRequest, ReceiveMessageResult>() { @Override`	`CompletableFuture<ReceiveMessageResponse> completableFuture = sqsAsyncBatchManager .receiveMessage(ReceiveMessageRequest.builder() .queueUrl(queueUrl).build())`

변경 사항	v1	v2

```
            public void
 onSuccess(ReceiveM
essageRequest
 request,

          ReceiveMe
ssageResult result) {

 List<Message> messages
 = result.getMessages
();

 System.out.println
("Received " +
 messages.size() + "
 messages");
               for
 (Message message :
 messages) {

 System.out.println
("Message ID: " +
 message.getMessage
Id());

 System.out.println
("Body: " + message.g
etBody());
               }
            }

            @Override
            public void
 onError(Exception e) {

 System.err.println
("Error receiving
 messages: " + e.getMess
age());

 e.printStackTrace();
            }
```

```
        .whenComp
lete((receiveMessa
geResponse, throwable
) -> {
            if (throwabl
e != null) {

 System.err.println
("Error receiving
 messages: " + throwable
.getMessage());

 throwable.printSta
ckTrace();
            } else {

 List<Message> messages
 = receiveMessageResp
onse.messages();

 System.out.println
("Received " +
 messages.size() + "
 messages");
               for
 (Message message :
 messages) {

 System.out.println
("Message ID: " +
 message.messageId());

 System.out.println
("Body: " + message.b
ody());
               }
            }
        });
```

변경 사항	v1	v2
	});	

주요 구성 파라미터

파라미터	v1	v2
최대 배치 크기	maxBatchSize (배치당 기본 요청 10개)	maxBatchSize (배치당 기본 요청 10개)
배치 대기 시간	maxBatchOpenMs (기본값 200ms)	sendRequestFrequency (기본값 200ms)
표시 제한 시간	visibilityTimeoutSeconds (대기열 기본값의 경우 -1)	receiveMessageVisibilityTimeout (기본 대기열)
최소 대기 시간	longPollWaitTimeoutSeconds (longPoll이 true인 경우 20초)	receiveMessageMinWaitDuration (기본값 50ms)
메시지 속성	ReceiveMessageRequest 를 사용하여 설정	receiveMessageAttributeNames (기본적으로 없음)
시스템 속성	ReceiveMessageRequest 를 사용하여 설정	receiveMessageSystemAttributeNames (기본적으로 없음)
긴 폴링	longPoll(기본값 true)	서버가 메시지를 보낼 때까지 대기 중인 열린 연결을 방지하기 위해 지원되지 않음
긴 폴링의 최대 대기 시간	longPollWaitTimeoutSeconds (기본값 20초)	서버가 메시지를 보낼 때까지 대기 중인 열린 연결을 방지하기 위해 지원되지 않음

파라미터	v1	v2
클라이언트 측에서 미리 가져와서 저장하는 수신 배치의 최대 개수	`maxDoneReceiveBatches` (배치 10개)	내부적으로 처리되므로 지원되지 않음
동시에 처리되는 최대 활성 아웃바운드 배치 수	`maxInflightOutboundBatches` (기본값 배치 5개)	내부적으로 처리되므로 지원되지 않음
동시에 처리되는 최대 활성 수신 배치 수	`maxInflightReceiveBatches` (기본값 배치 10개)	내부적으로 처리되므로 지원되지 않음

Java용 SDK 1.x와 2.x를 나란히 사용

프로젝트에서 AWS SDK for Java의 두 버전을 모두 사용할 수 있습니다.

다음은 버전 1.x의 Amazon S3와 버전 2.27.21의 DynamoDB을 사용하는 프로젝트의 `pom.xml` 파일 예제입니다.

Example POM의 예제

이 예제는 SDK 1.x 및 2.x 버전을 모두 사용하는 프로젝트의 `pom.xml` 파일 항목을 보여줍니다.

```
<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>2.27.21</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>
```

AWS SDK for Java 1.x 클라이언트를 사용하여 애플리케이션 찾기

AWS SDK for Java 2.x로 마이그레이션하기 전에 환경에서 버전 1.x 클라이언트를 사용하는 애플리케이션을 식별해야 합니다. AWS CloudTrail 이벤트의 'userAgent' 필드를 쿼리하여 이 애플리케이션을 찾을 수 있습니다.

CloudTrail Lake를 사용하여 1.x 클라이언트가 있는 애플리케이션 찾기

AWS CloudTrail Lake를 사용하면 CloudTrail에서 기록한 이벤트를 쿼리할 수 있습니다. 다음 단계에 따라 애플리케이션에서 사용하는 SDK 버전을 식별하는 데이터 레이크를 만듭니다.

1. CloudTrail 데이터 레이크를 만듭니다. 이벤트 데이터 스토어를 만들려면 사용 설명서를 참조하세요.
2. 데이터 스토어를 만든 후 레코드 콘텐츠를 검사합니다. 레코드 본문에는 요청된 작업, 타이밍 및 위치를 결정하는 필드가 포함되어 있습니다. 자세한 내용은 CloudTrail 레코드 콘텐츠에 대한 사용 설명서를 참조하세요.
3. 데이터에 대해 쿼리를 실행합니다. 사용 설명서에 따라 쿼리하고 쿼리 결과를 저장합니다.

각 레코드의 userAgent 필드에는 요청된 SDK 버전이 포함되어 있습니다. 이 필드를 사용하여 Java SDK의 버전 1.x를 사용하는 애플리케이션을 식별합니다.

다음 샘플 쿼리는 EventDatastoreID(`sample-Data-Store-Id`)에서 2025년 6월 17일부터 Java SDK 1.x로 이루어진 모든 요청을 찾습니다.

```
select userIdentity, eventSource, awsRegion,
    eventName, eventType, eventTime, userAgent,
    requestParameters, sourceIPAddress
```

```
 from sample-Data-Store-Id
where eventTime > '2025-06-17 00:00:00'
and userAgent like '%aws-sdk-java/1.%'
order by eventTime desc
```

쿼리 결과의 이벤트 콘텐츠 예제는 다음과 같습니다.

```
{
    "userIdentity": "{
         "type": "IAMUser",
         "principalId": "AIDAJ45Q7YFFAREXAMPLE",
         "arn": "arn:aws:iam::123456789012:user/Alice",
         "accountId": "123456789012",
         "accessKeyId": "",
         "userName": "Alice"
    }",
    "eventSource": "dynamodb.amazonaws.com",
    "awsRegion": "us-west-2",
    "eventName": "ListTables",
    "eventType": "AwsApiCall",
    "eventTime": "2025-07-01 02:23:52.000",
    "userAgent": "aws-sdk-java/1.12.746 Linux/5.10.240 OpenJDK/11.0.25+9-LTS ...",
    "requestParameters": "",
    "sourceIPAddress": "12.345.6.78"
}
```

이 정보를 사용하여 요청이 이루어진 때와 위치를 확인할 수 있습니다.

이 예제에서 DynamoDB `ListTables` 요청은 Alice라는 이름의 IAM 사용자 자격 증명을 사용하여 IP 주소(`12.345.6.78`)에서 `2025-07-01 02:23:52 (UTC)`에 생성됐습니다. userAgent 필드의 값은 요청이 JDK 11이 설치된 Linux 시스템의 AWS SDK for Java 버전 `1.12.746`을 사용하여 이루어졌음을 보여줍니다.

AWS CloudTrail 이벤트 레코드의 필드에 대한 설명은 관리, 데이터 및 네트워크 활동 이벤트에 대한 CloudTrail 레코드 콘텐츠를 참조하세요.

에 대한 보안 AWS SDK for Java

Amazon Web Services(AWS)에서 가장 우선순위가 높은 것이 클라우드 보안입니다. AWS 고객으로서 여러분은 가장 높은 보안 요구 사항을 충족하기 위해 설계된 데이터 센터 및 네트워크 아키텍처의 혜택을 받게 됩니다. 보안은 AWS 와 사용자 간의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드 내 보안 및 클라우드의 보안으로 설명합니다.

클라우드 보안 - AWS 는 클라우드에서 제공되는 모든 서비스를 실행하는 인프라를 보호하고 안전하게 사용할 수 있는 서비스를 AWS 제공할 책임이 있습니다. 당사의 보안 책임은에서 최우선 순위이며 AWS, 타사 감사자는 AWS 규정 준수 프로그램의 일환으로 보안의 효과를 정기적으로 테스트하고 검증합니다.

클라우드의 보안 - 사용자의 책임은 사용 중인 AWS 서비스와 데이터의 민감도, 조직의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요인에 따라 결정됩니다.

이 AWS 제품 또는 서비스는 지원하는 특정 Amazon Web Services(AWS) 서비스를 통해 공동 책임 모델을 따릅니다. AWS 서비스 보안 정보는 AWS 서비스 보안 설명서 페이지 및 규정 AWSAWS 준수 프로그램의 규정 준수 작업 범위에 속하는 서비스를 참조하세요.

주제

in AWS SDK for Java 2.x의 데이터 보호

AWS 공동 책임 모델의 데이터 보호에 적용됩니다 AWS SDK for Java. 이 모델에 설명된 대로 AWS 는 모든를 실행하는 글로벌 인프라를 보호할 책임이 있습니다 AWS 클라우드. 사용자는 이 인프라에 호스팅되는 콘텐츠에 대한 통제 권한을 유지할 책임이 있습니다. 사용하는 AWS 서비스 의 보안 구성과 관리 태스크에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 관한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

데이터 보호를 위해 자격 증명을 보호하고 AWS 계정 AWS IAM Identity Center 또는 AWS Identity and Access Management (IAM)를 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용합니다.
- SSL/TLS를 사용하여 AWS 리소스와 통신합니다. TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다 AWS CloudTrail. CloudTrail 추적을 사용하여 AWS 활동을 캡처하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 추적 작업을 참조하세요.
- 내의 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용합니다 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용합니다.
- 명령줄 인터페이스 또는 API를 AWS 통해에 액세스할 때 FIPS 140-3 검증 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 연방 정보 처리 표준(FIPS) 140-3을 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 형식 텍스트 필드에 입력하지 않는 것이 좋습니다. 여기에는 SDK for Java 또는 기타 AWS 서비스 에서 콘솔 AWS CLI, API 또는 AWS SDKs를 사용하여 작업하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 자격 증명을 URL에 포함해서는 안 됩니다.

Java용 SDK에서 TLS 작업

는 기본 Java 플랫폼의 TLS 기능을 AWS SDK for Java 사용합니다. 이 항목에서는 Amazon Corretto 17에서 사용하는 OpenJDK 구현을 사용한 예제를 보여줍니다.

를 사용하려면 기본 JDK AWS 서비스가 TLS 1.2의 최소 버전을 지원해야 하지만 TLS 1.3이 권장됩니다.

사용자는 SDK와 함께 사용 중인 Java 플랫폼의 설명서를 참조하여 기본적으로 활성화된 TLS 버전과 특정 TLS 버전을 활성화 및 비활성화하는 방법을 확인해야 합니다.

TLS 버전 정보 확인 방법

OpenJDK를 사용하는 다음 코드는 SSLContext를 사용하여 지원되는 TLS/SSL 버전을 출력하는 방법을 보여줍니다.

```
System.out.println(Arrays.toString(SSLContext.getDefault().getSupportedSSLParameters().getProto
```

예를 들어, Amazon Corretto 17(OpenJDK)은 다음과 같은 출력을 생성합니다.

```
[TLSv1.3, TLSv1.2, TLSv1.1, TLSv1, SSLv3, SSLv2Hello]
```

작동 중인 SSL 핸드셰이크와 사용된 TLS 버전을 보려면 시스템 속성 javax.net.debug를 사용하면 됩니다.

예를 들어, TLS를 사용하는 자바 애플리케이션을 실행해 보세요.

```
java app.jar -Djavax.net.debug=ssl:handshake
```

애플리케이션은 다음과 유사한 SSL 핸드셰이크를 기록합니다.

```
...
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",
...
```

최소 TLS 버전 적용

Java용 SDK는 항상 플랫폼 및 서비스에서 지원하는 최신 TLS 버전을 선호합니다. 특정 최소 TLS 버전을 적용하려면 Java 플랫폼 설명서를 참조하세요.

OpenJDK 기반 JVM의 경우 시스템 속성 `jdk.tls.client.protocols`을 사용할 수 있습니다.

예를 들어, TLS 1.3을 사용할 수 있더라도 애플리케이션의 SDK 서비스 클라이언트가 TLS 1.2를 사용하도록 하려면 다음 시스템 속성을 제공하세요.

```
java app.jar -Djdk.tls.client.protocols=TLSv1.2
```

AWS API 엔드포인트를 TLS 1.2로 업그레이드

최소 버전의 TLS 1.2로 이동하는 AWS API 엔드포인트에 대한 자세한 내용은이 블로그 게시물을 참조하세요.

자격 증명 및 액세스 관리

AWS Identity and Access Management (IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어하는 데 도움이 AWS 서비스 되는 입니다. IAM 관리자는 누가 AWS 리소스를 사용할 수 있는 인증(로그인) 및 권한(권한 있음)을 받을 수 있는지 제어합니다. IAM은 추가 비용 없이 사용할 수 AWS 서비스 있는 입니다.

주제

- 대상
- ID를 통한 인증
- 정책을 사용하여 액세스 관리
- IAM AWS 서비스 작업 방법
- AWS 자격 증명 및 액세스 문제 해결

대상

AWS Identity and Access Management (IAM)를 사용하는 방법은에서 수행하는 작업에 따라 다릅니다 AWS.

서비스 사용자 - AWS 서비스 를 사용하여 작업을 수행하는 경우 필요한 자격 증명과 권한을 관리자가 제공합니다. 더 많은 AWS 기능을 사용하여 작업을 수행하게 되면 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방법을 이해하면 관리자에게 올바른 권한을 요청하는 데 도움이 됩니다. 에서 기능에 액세스할 수 없는 경우 사용 중인의 AWS 자격 증명 및 액세스 문제 해결 또는 사용 설명서를 AWS참조 AWS 서비스 하세요.

서비스 관리자 - 회사에서 AWS 리소스를 책임지고 있는 경우에 대한 전체 액세스 권한을 가지고 있을 것입니다 AWS. 서비스 관리자는 서비스 사용자가 액세스해야 하는 AWS 기능과 리소스를 결정합니

다. 그런 다음 IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해하세요. 회사가 IAM을 사용하는 방법에 대해 자세히 알아보려면 사용 중인의 AWS 서비스 사용 설명서를 AWS참조하세요.

IAM 관리자 - IAM 관리자라면 AWS에 대한 액세스 권한 관리 정책 작성 방법을 자세히 알고 싶을 것입니다. IAM에서 사용할 수 있는 자격 AWS 증명 기반 정책 예제를 보려면 사용 중인의 사용 설명서를 참조 AWS 서비스 하세요.

ID를 통한 인증

인증은 자격 증명 자격 증명을 AWS 사용하여에 로그인하는 방법입니다. AWS 계정 루트 사용자, IAM 사용자 또는 IAM 역할을 수임하여 인증해야 합니다.

AWS IAM Identity Center (IAM Identity Center), Single Sign-On 인증 또는 Google/Facebook 자격 증명과 같은 자격 증명 소스의 자격 증명을 사용하여 페더레이션 자격 증명으로 로그인할 수 있습니다. 로그인하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의 AWS 계정에 로그인하는 방법 섹션을 참조하세요.

프로그래밍 방식 액세스를 위해는 요청에 암호화 방식으로 서명할 수 있는 SDK 및 CLI를 AWS 제공합니다. 자세한 내용은 IAM 사용자 설명서의 API 요청용AWS Signature Version 4 섹션을 참조하세요.

AWS 계정 루트 사용자

를 생성할 때 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 AWS 계정 theroot 사용자라는 하나의 로그인 자격 증명으로 AWS 계정시작합니다. 일상적인 태스크에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 자격 증명이 필요한 작업은 IAM 사용자 설명서의 루트 사용자 자격 증명이 필요한 작업 섹션을 참조하세요.

페더레이션 ID

가장 좋은 방법은 인간 사용자에게 자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 AWS 서비스 사용하여에 액세스하도록 요구하는 것입니다.

페더레이션 자격 증명은 엔터프라이즈 디렉터리, 웹 자격 증명 공급자 또는 자격 증명 소스의 자격 증명을 AWS 서비스 사용하여 Directory Service 에 액세스하는 사용자입니다. 페더레이션 ID는 임시 자격 증명을 제공하는 역할을 수임합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center를 추천합니다. 자세한 정보는 AWS IAM Identity Center 사용 설명서의 What is IAM Identity Center?를 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가진 ID입니다. 장기 자격 증명이 있는 IAM 사용자 대신 임시 자격 증명을 사용하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 AWS 사용하여에 액세스하도록 인간 사용자에게 요구하기를 참조하세요.

IAM 그룹은 IAM 사용자 모음을 지정하고 대규모 사용자 집합에 대한 관리 권한을 더 쉽게 만듭니다. 자세한 내용은 IAM 사용자 설명서의 IAM 사용자 사용 사례 섹션을 참조하세요.

IAM 역할

IAM 역할은 임시 자격 증명을 제공하는 특정 권한이 있는 자격 증명입니다. 사용자에서 IAM 역할(콘솔)로 전환하거나 또는 API 작업을 호출하여 역할을 수임할 수 있습니다. AWS CLI AWS 자세한 내용은 IAM 사용자 설명서의 역할 수임 방법을 참조하세요.

IAM 역할은 페더레이션 사용자 액세스, 임시 IAM 사용자 권한, 교차 계정 액세스, 교차 서비스 액세스 및 Amazon EC2에서 실행되는 애플리케이션에 유용합니다. 자세한 내용은 IAM 사용자 설명서의 교차 계정 리소스 액세스를 참조하세요.

정책을 사용하여 액세스 관리

정책을 AWS 생성하고 자격 증명 또는 리소스에 연결하여 AWS 에서 액세스를 제어합니다. 정책은 자격 증명 또는 리소스와 연결될 때 권한을 정의합니다.는 보안 주체가 요청할 때 이러한 정책을 AWS 평가합니다. 대부분의 정책은에 JSON 문서 AWS 로 저장됩니다. JSON 정책 문서에 대한 자세한 내용은 IAM 사용자 설명서의 JSON 정책 개요 섹션을 참조하세요.

정책을 사용하여 관리자는 어떤 보안 주체가 어떤 리소스에 대해 어떤 조건에서 작업을 수행할 수 있는지 정의하여 누가 무엇을 액세스할 수 있는지 지정합니다.

기본적으로 사용자 및 역할에는 어떠한 권한도 없습니다. IAM 관리자는 IAM 정책을 생성하고 사용자가 수임할 수 있는 역할에 추가합니다. IAM 정책은 작업을 수행하기 위해 사용하는 방법과 관계없이 작업에 대한 권한을 정의합니다.

자격 증명 기반 정책

ID 기반 정책은 ID(사용자, 사용자 그룹 또는 역할)에 연결하는 JSON 권한 정책 문서입니다. 이러한 정책은 자격 증명이 수행할 수 있는 작업, 대상 리소스 및 이에 관한 조건을 제어합니다. ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용자 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

ID 기반 정책은 인라인 정책(단일 ID에 직접 포함) 또는 관리형 정책(여러 ID에 연결된 독립 실행형 정책)일 수 있습니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용자 설명서의 관리형 정책 및 인라인 정책 중에서 선택 섹션을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 예를 들어 IAM 역할 신뢰 정책 및 Amazon S3 버킷 정책이 있습니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 IAM의 AWS 관리형 정책을 사용할 수 없습니다.

액세스 제어 목록(ACL)

액세스 제어 목록(ACL)은 어떤 위탁자(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

Amazon S3 AWS WAF 및 Amazon VPC는 ACLs. ACL에 관한 자세한 내용은 Amazon Simple Storage Service 개발자 가이드의 액세스 제어 목록(ACL) 개요를 참조하세요.

기타 정책 타입

AWS 는 보다 일반적인 정책 유형에서 부여한 최대 권한을 설정할 수 있는 추가 정책 유형을 지원합니다.

- 권한 경계 - ID 기반 정책에서 IAM 엔터티에 부여할 수 있는 최대 권한을 설정합니다. 자세한 정보는 IAM 사용자 설명서의 IAM 엔터티의 권한 범위를 참조하세요.
- 서비스 제어 정책(SCP) - AWS Organizations내 조직 또는 조직 단위에 대한 최대 권한을 지정합니다. 자세한 내용은AWS Organizations 사용 설명서의 서비스 제어 정책을 참조하세요.
- 리소스 제어 정책(RCP) – 계정의 리소스에 사용할 수 있는 최대 권한을 설정합니다. 자세한 내용은 AWS Organizations 사용 설명서의 리소스 제어 정책(RCP)을 참조하세요.
- 세션 정책 – 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 자세한 내용은 IAM 사용자 설명서의 세션 정책을 참조하세요.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 에서 여러 정책 유형이 관련될 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하세요.

IAM AWS 서비스 작업 방법

가 대부분의 IAM 기능을 AWS 서비스 사용하는 방법을 개괄적으로 알아보려면 IAM 사용 설명서의 AWS IAM으로 작업하는 서비스를 참조하세요.

IAM AWS 서비스 에서 특정를 사용하는 방법을 알아보려면 관련 서비스 사용 설명서의 보안 섹션을 참조하세요.

AWS 자격 증명 및 액세스 문제 해결

다음 정보를 사용하여 및 IAM으로 작업할 때 발생할 수 있는 일반적인 문제를 진단 AWS 하고 수정할 수 있습니다.

주제

- 에서 작업을 수행할 권한이 없음 AWS
- iam:PassRole을 수행하도록 인증되지 않음
- 내 외부의 사람이 내 AWS 리소스에 액세스 AWS 계정 하도록 허용하고 싶습니다.

에서 작업을 수행할 권한이 없음 AWS

작업을 수행할 권한이 없다는 오류가 표시되면 작업을 수행할 수 있도록 정책을 업데이트해야 합니다.

다음의 예제 오류는 mateojackson IAM 사용자가 콘솔을 사용하여 가상 *my-example-widget* 리소스에 대한 세부 정보를 보려고 하지만 가상 awes:*GetWidget* 권한이 없을 때 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 awes:GetWidget on resource: my-example-widget
```

이 경우, awes:*GetWidget* 작업을 사용하여 *my-example-widget* 리소스에 액세스할 수 있도록 mateojackson 사용자 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

iam:PassRole을 수행하도록 인증되지 않음

`iam:PassRole` 작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 AWS에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부 AWS 서비스 에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 기존 역할을 해당 서비스에 전달할 수 있습니다. 이렇게 하려면 역할을 서비스에 전달할 권한이 있어야 합니다.

다음 예 오류는 `marymajor`라는 IAM 사용자가 콘솔을 사용하여 AWS에서 작업을 수행하려고 하는 경우에 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 권한이 없습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우, Mary가 `iam:PassRole`작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

내 외부의 사람이 내 AWS 리소스에 액세스 AWS 계정 하도록 허용하고 싶습니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우, 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세한 내용은 다음을 참조하세요.

- 에서 이러한 기능을 AWS 지원하는지 여부를 알아보려면 섹션을 참조하세요IAM AWS 서비스 작업 방법.
- 소유 AWS 계정 한의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 소유한 다른의 IAM 사용자에게 액세스 권한 제공을 참조 AWS 계정 하세요.
- 타사에 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사가 AWS 계정 소유한에 대한 액세스 권한 제공을 AWS 계정참조하세요.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용자 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(ID 페더레이션)을 참조하세요.
- 크로스 계정 액세스에 대한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용자 설명서의 IAM의 크로스 계정 리소스 액세스를 참조하세요.

이 AWS 제품 또는 서비스에 대한 규정 준수 검증

AWS 서비스 가 특정 규정 준수 프로그램의 범위 내에 있는지 알아보려면 AWS 서비스 규정 준수 프로그램 범위 내 참조하고 관심 있는 규정 준수 프로그램을 선택합니다. 일반 정보는 AWS 규정 준수 프로그램.

를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다 AWS Artifact. 자세한 내용은 Downloading Reports inDownloading AWS Artifact을 참조하세요.

사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표, 관련 법률 및 규정에 따라 AWS 서비스 결정됩니다. 사용 시 규정 준수 책임에 대한 자세한 내용은 AWS 보안 설명서를 AWS 서비스참조하세요.

이 AWS 제품 또는 서비스는 지원하는 특정 Amazon Web Services(AWS) 서비스를 통해 공동 책임 모델을 따릅니다. AWS 서비스 보안 정보는 AWS 서비스 보안 설명서 페이지 및 규정 AWSAWS 준수 프로그램의 규정 준수 작업 범위에 속하는 서비스를 참조하세요.

이 AWS 제품 또는 서비스에 대한 복원력

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다.

AWS 리전 는 지연 시간이 짧고 처리량이 많으며 중복성이 높은 네트워킹과 연결된 물리적으로 분리되고 격리된 여러 가용 영역을 제공합니다.

가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 극복 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하세요.

이 AWS 제품 또는 서비스는 지원하는 특정 Amazon Web Services(AWS) 서비스를 통해 공동 책임 모델을 따릅니다. AWS 서비스 보안 정보는 AWS 서비스 보안 설명서 페이지 및 규정 AWSAWS 준수 프로그램의 규정 준수 작업 범위에 속하는 서비스를 참조하세요.

이 AWS 제품 또는 서비스에 대한 인프라 보안

이 AWS 제품 또는 서비스는 관리형 서비스를 사용하므로 글로벌 네트워크 보안으로 AWS 보호됩니다. AWS 보안 서비스 및가 인프라를 AWS 보호하는 방법에 대한 자세한 내용은 AWS 클라우드 보

안을 참조하세요. 인프라 보안 모범 사례를 사용하여 AWS 환경을 설계하려면 보안 원칙 AWS Well-Architected Framework의 인프라 보호를 참조하세요.

AWS 에서 게시한 API 호출을 사용하여 네트워크를 통해이 AWS 제품 또는 서비스에 액세스합니다. 클라이언트는 다음을 지원해야 합니다.

- Transport Layer Security(TLS). TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군. Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 위탁자와 관련된 시크릿 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용하여 임시 보안 자격 증명을 생성하여 요청에 서명할 수 있습니다.

이 AWS 제품 또는 서비스는 지원하는 특정 Amazon Web Services(AWS) 서비스를 통해 공동 책임 모델을 따릅니다. AWS 서비스 보안 정보는 AWS 서비스 보안 설명서 페이지 및 규정 AWSAWS 준수 프로그램의 규정 준수 노력 범위에 속하는 서비스를 참조하세요.

AWS SDK for Java용 OpenPGP 키

AWS SDK for Java에 대해 공개적으로 사용 가능한 모든 Maven 아티팩트는 OpenPGP 표준을 사용하여 서명됩니다. 아티팩트의 서명을 확인하는 데 필요한 공개 키는 다음 섹션에서 확인할 수 있습니다.

현재 키

다음 표는 현재 릴리즈의 SDK for Java 1x 및 SDK for Java 2.x에 대한 OpenPGP 키 정보를 보여줍니다.

키 ID	0xAC107B386692DADD
유형	RSA
크기	4096/4096
생성 완료	2016-06-30
Expires	2026-09-27
사용자 ID	AWS SDK 및 도구 <aws-dr-tools@amazon.com>
키 지문	FEB9 209F 2F2F 3F46 6484 1E55 AC10 7B38 6692 DADD

SDK for Java의 다음 OpenPGP 공개 키를 클립보드에 복사하려면 오른쪽 상단 모서리에 있는 "복사" 아이콘을 선택합니다.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Comment: Hostname:
Version: Hockeypuck 2.2

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
zSxBV1MgU0RLcyBhbmQgVG9vbHMgPGF3cy1kci10b29sc0BhbWF6b24uY29tPsLB
lAQTAQoAPgIbAwULCQgHAwUVCgkICwUWAgMBAAIeAQIXgBYhBP65IJ8vLz9GZIQe
VawQezhmktrdBQJo12ZrBQkTQxnmAAoJEKwQezhmktrdi18P/A3De83MBx8bdcWJ
Fot71Vk1TyBQFErgrtcytSU0czEHx3tGbzgQLbMlyzjirOT03usxEkOeqTVK+RU+
5uFXNZYQLwMJlHJ6S8tnfLe/ExM5WQ2KPwIUPfZs1GDDRQB2dIKSc+qYrP1O1vf4
O4iPgfLHMW2bFh3zjjxcaHCJyqc7Cau33eZFBAsRni1jOUo7MeyX0hlXfW8pd48Q
wZllQVZ/6KmDiFWA0CZ+2svJ5cL0tgPoh1OQjoz0nHpNfuDILMrZ+e7tx2VTlkGH
UGeNSydnrK8v9ztFn34KtU/k7NEWoVSyEi+5ICZL18FBwPqTwdVWXwXrqZCKiIpr
8ZdJWDz2sJfgDFNCC6rKgCQ6FrmaD9G76dYWkQ4AbZqABlUzU3q36W1K0r3iOAb5
G4tdOt4yqXHTe1x+ZUNaeW7gaCmtXAxLw0OfeJrcq/44b/SQP+qJ8sSOv76Yg2oF
BsF5DWOVUFghbTyokHAoVROyhBR4dUUisY39AqLSL8+Lp9Pr3wNuGl9GLrMD5701
piUb88B3Gwe1EiKV1gaKrvZ3mECDUiSMVO0Z5iG8E4QDpNmVbJbV1uT821ubvtOv
2Ko10Fa0uwCYGssdRGqEXNy6jz/Er8LAC3+nmGINDJQzrF+loYoSSkI2Nu7lhMuL
7iWwUPF7OhDXoVSAn4X3x6q2rGK0wsGUBBMBCgA+AhsDBQsJCAcDBRUKCQgLBRYC
AwEAAh4BAheAFiEE/rkgny8vP0ZkhB5VrBB7OGaS2t0FAmjXZTsFCRNDGLYACgkQ
rBB7OGaS2t0/Ow//YIv51vHtD+kwMmIvk3zpizDHY0zW2dOezAo+C/DsSyC7wDll
Dixw34EQ1yLXH5xLR8CH1zupl3JmmEp1ucdQggoefbidxDl8Fld7tJOD1y3GGnTD
0jAl2ZC+W65Oh+wS1mD1FlaKjMGGkvJf0dA7RtU2T8dv3vt8dsxg76FMFS3+fqlC
FNOAsNTn9zWR1SqBIfkMJK83aq6s/rcEV9VrAYgDgqex58fygB5EuTf842/IF7WZ
Q9gd6fupB0mMZP5YWd2uj/vsBTYakG+mgQwDxZuKPeEzAqnqqS7biSQOUO6Wozlq
Yy4fSczE9GkBAvg0pGmbko+zHvpnjvX/h1CUpC6odvFyOAhZp6zyhs0QWz9thfqV
lU8WlbgJ2atFDn5GUSxF/fe0Yzovlbbs6sbYXuvMG9RiEOuJ1mBbZR3aIdZ1U6Do
BHc/vjc5mWcV7JQSP7i4W/8W7X3UAuN9LdxB+IvF3Cwrgtlw2BWvA5Alco5Tnz8t
P/CIVmBjk+sLme8W4kfLK3IWEbwClOdNnErI/MHRm65A2Y5EMIhwjrOi07SU1Pxa
nPpg3OYJCdvjzdB8QE3/DBiMfOl4dISfKDVEWnfK8mZaYd/BeRm2gUAa9UrqSFCG
BlA7Lg+eLI3US0FvwWJ4j5bBJqgLu+y7crIkiUOPAQuLk3lO+5uYU/I3DuLCwZQE
EwEKAD4CGwMFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AWIQT+uSCfLy8/RmSEHlWs
EHs4ZpLa3QUCZwAXCAUJEWvKgwAKCRCsEHs4ZpLa3ZdTEACMBLg2q9zk8ZH02nDz
Sg5zc8Wlqq8WdxU0Pj8qx4UOrrMca7wyiUvrgoxPW5lh1RVNUeMkDRfu9pSXcOVI
V9LvmYE/WnwKROubgGbsC4T7M/LqV0/AulXil4d7IXcO6l4toa8LTNWtD5bODgrN
gvay1AzCU8kq1Qw1cKZ2gAfvA3Ba7PWyLeUN4HTlGrXcw73G+0CofY1L8wqWxHCJ
29XqQzeTEc6MDEeIlNlVdUcy8Qr5uwkEsl34H9AxS5F1opJ4TqvXiDZsrSRRv57R
XYmRZDWeYT+9PZaMsHXza5qgej7BfATxhYfICsNaY6MK3x6b+nDSKkoZgO+i09zh
1YjpahhQe6G336v/3mRj0dKGCRQ6znQ9ghUaB5z9zfvgH5AOEkTe3l8MqM+j5A6P
VjSBBJAHKejxr7+wKJKIA6P+DqpsYAunzftwUzrLVqb+BZQ+DcTmVrE7OPcMYJD5
QglX/Le+WmWZHI154NXgpWWUOUgZUbUge4DKrT+zCJ9iecPLKTW7OcULyXO+rjb8
8BGrD5GPlHB3dOUXXTlMKCqg3qy1Bu2KnZTQiaEEdZgSIGQbrW0JTMmmXJkKjokd
JMA4vYeg5en51G9nRQjScPngx77IxvByNyFWTJdG1ENpJpsK9TtmENcpyUJtJZTJ
ZSOIRVPP5RzR5vInuXWq6VV0BMLBlAQTAQoAPgIbAwULCQgHAwUVCgkICwUWAgMB
```

```
AAIeAQIXgBYhBP65IJ8vLz9GZIQeVawQezhmktrdBQJlJEoiBQkPj/2dAAoJEKwQ
ezhmktrdx1YP/0vvym3jgX/pwnR7K1rafZMb1iKQBr0ISG8cdbaf4pqX5vuUZnyj
w9Cl/oONn7jJjnQxOIIzuBoxne2WN28ftM2w0nVXm85mAmz2fwQz/fdKDyonXcOh
pfD2iMqn7gESjhEgRE7wMDYMDuLdqHI7OKWGVfgrh7xEmKapLh45h7cnumo2VjL9
uDYY1aOBHz993T7oE41y43rhk+6kKbGFd2uuo7h5j1ZF8Lj6sYfcEzXOU1OhRlD0
nyBjDy9MYWu0YNouc70WgMceGx6hjvCAM/5fxP7SZFecZ7ePeB0GpvVA24hSNENE
0r3tUekuOf1I0FunMnMnbh7ZO9rPYqWvWDNIpU3S4CjFhY82L+IeKnmLy8N6ASRk
HsPiNCOHSK8C/0ynrd9xLhX8Jsk/TGiQYaleoHhWkNLlZsL86QHL8SKEqkqZCQf5
AEqghDP6NEGS7lnOenA7JjIrA9KLlT7fnNWZOwFi5X+o/CymE2ytEMS0Yf3nmY4U
n9x56Wgn6J2zqB5nqOXf6NxGdAIgOBm098YEnKCIFzk+yhoDlprVpHcnd2b5f6Oq
uh8KYOEbKgpMJ3zZuWSL5kwGF1nNoYiAkonMaz9H3pOQnOMVYCUeUTDRsiO/prrd
UhNlry4TAsBMpeXnFhdLVM3vFQZVpByadGOJNmnaN/Wavw2a00UGBFa4wsF9BBMB
CgAnAhsDBQsJCAcDBRUKCQgLBRYCAwEAAh4BAheABQJhMqGaBQkLnlUVAAoJEKwQ
ezhmktrd2sQP/3YHM+U+BbOy1nSEAfykZ71+uCM2hkHMLdxQYWB/rBWkmg/pbu+d
r4t45RsTASrNjRcZOntlPMQRIq973ymHfpmeS+noFwvTGH7zDv1BRBR9wPrd1XUz
iSuEUHGi/fqxUVXQ5mbonzfThX8tuXeuiQmeToqoB00FYlZm6xsNnEHcjVl66mC4
IPoJLWnZJs4rOCeoRf5XvDTgX6xt5/kLYRZf79qaWGFvaZpsc1CH+rQJUdVa/D4T
7pI7hX6zy0S91z4iuC5HZUiOTF+y5auEZHGTdTWNS1kvOvfcCTi0XK/GkGL82SZu
7X2VGnpCeUnFyViRGlk+KaDG1sVyDY+lcBPg6ilr45M6MQV0iHS5OF04QNXSKt5+
UnzJH7lldgNsR6ibRMyNV3k5v3fyUcSBvIYyLORTTBiVEjQDSbk1QNqbrQlX9CWz
+EJWn16BFTmMFvxBSWPm640GncHP5J3/0MbMw3Cm90x7k8UfNANIemcrJrSxIDwm
g9cVAg3a+D+wxjrVe8jGg0ejvECpm+0yswigj5x6Lqj09A4UgdjEauN+/pn0nhBo
Gv7DzMXtM/LoDtgp6wn93qZVN2TsuHnkEk4UyntB6eWJbBdXHWUr47exiWh0dvQN
tpwCWPT6I7ZTPtA5K/zx+q9m6797BLgAkTYc6gloQL3vs1Z1S3m/hZNawsF9BBMB
CgAnAhsDBQsJCAcDBRUKCQgLBRYCAwEAAh4BAheABQJgmrz2BQkLBnBxAAoJEKwQ
ezhmktrd36oP/2rB2EkwSOCKC4m0heWSfDWi6OBKoEbbDtFtc6/HwqBW8SPsiKlq
zV0e3qBY/LVju04+ktJEK+EGXLnC3iC36MegrQ8zt391kEx/Zv9LIuVOCX90QIAX
dL8MVUkkjRLCFFH8pTgRy1cJYWk1X4dYdXWYc29fCwNVartNdNBhsb2ht3VJeKDE
kUivBHmkjuISDPEnI1coY7Lj0ZtY5cHdRF2eZpB0RkTBpsIt18rCYyHkERZrhmvb
j3rOyPyvOa+1/dQS8/hv5pEmbKx8cy8RdJkmbUHYatPBsjHkJSWr7O7G9VFW4GoN
9CRAI4KkbDSEDjCL5dv2pq0Sew1MkLuWJGULAMgiIUlWcOs5SZZGFSksNQrtSFV9
Z/wGocecMGkGQNXQ06JV/Fry/TvyphBlmylEqL+NLqEcEjnlz90IVu+ZA+M09J96
UlHO7V5GvBgM+QK/q/dJeMHPWrNlo1gA6Nwl/HBdM0DqzdZ2jEPvsQSABvZrPMty
+BAqEar4wqY1AH4X5ccEjO7nJQoBQSDRSki1fkBsc1nx44N/m0kHdIa0Z/Y+Mw4v
WiZhREkOospG1I4lBa3CNTVAhSs9msGsYfkqvFJGHL7sZY8XSv82GBBvA0nUNrsJ
bLBwo2FaQG9eoatRAGkqp4b/OtNtBuGeiQoNwFGbfUZTAaStj5/zZj0swsF9BBMB
CgAnAhsDBQsJCAcDBRUKCQgLBRYCAwEAAh4BAheABQJe+9bwBQkJZ4plAAoJEKwQ
ezhmktrd+ScP/RoaUKriVVAgLHOGs+/mnfKtnfTlClzi5dsdI9/6H0vLpmSWK/Cl
2cT6gary45VMgAeVK+H1lQXafYj+FY++I5kYoe2GrSvIXhpjaFAJyNf/dKleTsqR
Tm371i8b3FDYs5kvy2CnTbmHB8MsOGxck8/YHd1x+g8WpO2IgF89yYCSF3CAdxC3
6bHbs6Z3C3lcM/3SoWF+Yie2P8XeBMPCGp/BcjQzUcHF6G06TwDDYhixucUi6vEY
EH5JtOwVVQ7bubT8OFeOoJwVxlzYz4UoqxjKDWymarTzu03AUIT0PXPece94bJAK
mSh68ItQe3H8tSPMubERWz2tEV3lVkChDGXcC7BYQmxHseolxz/qzCtJ0iX9BvZR
dniZNeNJ/Cu8M2pDp47zdNFXzf/Q/sQ9pQlws22G2g119rWDneBku9n1vTP80/er
```

```
SB+VLTBjDiArlCY5y9+BG8wbscExJySoQxkB9j/nlMzPY5rgk0SyxsNj9GbqH+hr
EjS3/uacNwSLxGcOT2E9Teot5pfTEO6fQVq+35QhfAlP8c8jze01W/+u+wXu1Ui9
azRSzYtCHanGyyet6U1mlBpAkqkZzH6t3CA5czc9i6FbzjvFVZnbRUZIRzfISYew
lF5WqgTn2iYVdxagPRvLF5kjd696brGW9d5HwirCVGaK04VsXWlAblB9wsF9BBMB
CgAnBQJXdYAFAhsDBQkHhh+ABQsJCAcDBRUKCQgLBRYCAwEAAh4BAheAAAoJEKwQ
ezhmktrdWigP/3QWl7aO81BUWyby4HEhN4SdAoWGY/FLqO4mCtuplcnMgRUCSiL9
l2BSCTMCtUcdSWtYwOgSChN2mMsdi1U2FNR5HvNunYR/pFdqjfQurf1ZmKVeG5/4
uuKaOxMw9e8pK5uYAfs+O7gr8gu/f6/Drp7NZk3/yVKpf4WCY9oX9TA1q9O/11nN
cwS45U/d7YP+N1YM9cBXa1DnDcdfm0BlykzouAF0qd1Lwi/tmLENvybD3+2c2WsE
rlFZGSa5ZafO0tTIWXh5k6wh5FdRRycrnSyRK3B9N9+yaXfMQ0XpOypa8dqQEnCi
IsngDCJPxtTrhMWKhBFRUMzK/WZTDboTQSQDK+YVRrE4K8MtoZSKwZLV2r9O3TpX
kpbKsPVYmexerfdMeZfjZMF1bC7BmEs7jciH6JjbqAoAPnHzN0481aeNarINSViX
PQWr2mp9qShei2/RavLtx2ZNrvmGW72ZKpF8E3WWUDpBJqFVeGNRvOm3aZj8o/Hl
ewtNjcT4ouJfqlfKiULv+g7ANEMDLQTFDTg5twRdvmZlB7oTBsavf+LwxPIXhH32
IR7TX7VeicMMxmZnmZK2ANT/QBi3laf+ojVHvB+f6D74eLNq0Zqjfi/3UFNYsYjg
E+YgCqEUBpHbl61nOHwGOSsQwfap2uKKlzukD/KxH5SPBC3DYGBI+KCbzsFNBFd1
gAUBEAC8zNArPWb3dPMThL2xAY+fS60vXdB1SkOtYJpDWpFgvo0d+VQ+hV6XulGA
HAS6xG1WHysPT9KejIRSgLG+e9CaM5yhsxNa1WFGUM4Q9ESo3t+a75Go7xHIxgFj
C046/O6Vh3g9N/PREeuG8zkZ3H2v5fmD+ejyPgk4W9sFL00zjRiZD0FKVYR/j9ue
nEC/2NBcLuFy3q6cDfmCoDEOO62kXMnaGz3knzEK/X1SkcjsxRDq7zaQlQ1Kou+3
dICwy4x5SJQ8jl+eeeEvF2C2/dXmDohb57tqUwioohMUQkmCtvZgEHjypUwgp0MT
o25gWxkvJlSJKUOb6b1786WNySIzF2gxqlkkEmBl4RAssQkeXjrSmGwsMDyHNqyJ
eYFusl8sPaSpo+V2n0z+2B070Uq+wmf1S5A5FpegH0PZzzoNZo8I6QxaZje9YSZU
ijGmZIdEBleRVt3Svhi8MYlnasd4bW2RK1sr7plkBf8QRe6biiQRF3KDOSn5CbmX
pAcHJ1ZHzRRdkXZDNQC6vCJxsy13O0TrhJtAV1Yq347uyUbVi291ISVgroUVtprs
mHoEk5GoOTHbg9SCSt+xi/FiJQC+ubWmIGXoFKMR3UmhDnnzobKcbnbs/Hd981Fd
VghYYvq//gTAkJk0WxfGqO30wtXRndPOA0T+qhP3TE+LtGRJ+wARAQABwsF8BBgB
CgAmAhsMFiEE/rkgny8vP0ZkhB5VrBB7OGaS2t0FAmjXZm4FCRNDGegACgkQrBB7
OGaS2t3y5g/7BFXp/fdanzuQPToJTPen7AVwhLloKaiYhG3GjdXfMPLvu6UtaaGm
qynLolUNNooobptFqc1G9BKoAghQrta7CsDHtsQF2xyc3Mfu0gmpL/7X5a7sFIeJ
j08UjfweHx4DSG4LEZgNaAoWFjZltp4+8cqijkAHXt+r+1ayQG4VVHOWyXXqmSH4
9HqtbPcPyRzxdoVLeshZC9jmhHhhKqw/LwGyipWSOUKQDjWarBwdyhNmWCaLvxH1
ndMp4tq8DPGC3G4T9tYAbANrn7nKfZgHebMSzMw9kSp0L6QvwwTDjJyIWz85WyeH
WHeBysDaBOit3XDlehUew27y7N6a9hQSYjnXuwvre5mjDIOqJon/31R6ui2Z1y9P
a+bC11hbLXXh9tLCXRuoOt6thh9Cq5X1a76PPpEv30o3bpsb6l2hbrut1OKezwvK
l7txito/jfMiWfsZHA9O4SoM+8GnmVingHtZ805n1T4RddJvT/vaqplfI6zf7jmf
a69lALP420riFOQcwntNUM5tVmFUZsnFp2YRd4Ls7MiXVjtABahlSbb94l5WSVc0
jrOLDf94edvzk4R8i2Ob8CfVZNqEsTR6bHz8dT7Q+xQzEdjUujyyZY1UUl157Qeb
OsHjhCtuZYCI04X9hZ37nKnZXSxRlRDCnt5BEiyFu2WD1RscUe6PcVDCwXwEGAEK
ACYCGwwWIQT+uSCfLy8/RmSEHlWsEHs4ZpLa3QUCaNdlPQUJE0MYuAAKCRCsEHs4
ZpLa3XCpD/42DrcveE+q2ulrAIYPDlUlHiwIMejqBDRm6zmr1KSAeb4E6/MFcP4s
rXSSscMlrqG6NVynjNCXjD2YzWii68EwoXLJkgoD3r2ifzkV62EX2MIEeNZAVwuy
KNxorzmy6bhuWltRYNK/hITs2AG5orOk9ADEJ8PixKymrWlhesPaWX6Yhp9/tWaC
RHOSRiLbRVaJ+7sqT88urLmkV9Hqx949Zxv4+cgBVUGL6WXKsfWhHjbDMNJnozWB
```

```
SZaIJznLApOM8z+1DNrqUYyfR8SkF4IOVmg6HDzoyuseJJ8JvMAlkvT6F9VBq/iE
yeDYdEEQxwHwozKrEx5Ybxl5mntbqwCXy6kHSx2+/3RZWpZQ8K29YP9QEk0KeGF8
9Vap3jjNrX4u3cuRNQpeblQc4uFn3Nzaj+cVV4YzcRw94NifecXpujSvk8XU2ytJ
/JgMBxPIBKglN4eEMet9b4FRB5XeBdPAm19/LXyb4IlIipGNXlgNz/HCuBzidzHT
QQdqfA9rZVx1hwFr7AJCVqWaXVsx1oEAhKqpTtsLMyj594DvnRuwKw5Vse+1eydW
MIHYdbxmJccsTGIt/hsOpc8zfm+QYk5752jshhOKEBy+Ey3QZI1WbO547NOb2Hwr
Pgt7fw2NCKMPElSu98zmneFPhqNHf7L5urBe5gADj81E8lm6t/oVxcLBfAQYAQoA
JgIbDBYhBP65IJ8vLz9GZIQeVawQezhmktrdBQJnABcLBQkRa8qFAAoJEKwQezhm
ktrde3MP/13CLWp99XvRROrzD/bWOfWjAenT2PE/tYdOY9YcTQFbnIUhaVUDWAo3
pibR3D4u9LlY4olpGfJ7BTIHFa9myfpaVvmrNjueYI4omli24JQ/CKqNdY8Qzxz+
/QyiNK7Aw5cEBWIu84WGB1SsefWWT3rZe9YBb77gNcWHZ15pXTXrcgUxGY4808MC
I9YFWq8EA0iHawtFnmB3UFfClWt37Hy3PKvr1is3uG60+ULI8RQz3/+ZwSG8U+xt
b+I7H9+gITc1eFCb+tIwp5xWflyxcFXYk6UzOL7y3Fg2tIEuSNtIHUC9NDVobf6c
I0KAzZcMvKiPQiuBnVOjgDLmCZM5H6axj9x+gi4oVh6ea3HLqMzyjm5JkeCGgKWv
H0gD3yGEZDvcbavkQOle5T+4JefndKzCPrluX0iyx+oQiiOL8WieSSkSB6BsZcUN
SeuGJwM79Y7Oqld/YVrQNBZj5Vz+m3nZ+0EWDDMI0hRgMpSEIc+dnTC0u103Z+Rc
c2IJq8INmU653sUcfCZE12ParW4rF7ib6kViYrABT8f4e2TP0aOyP5kp51ied9qL
azaBA6tt/C9X1V2EJZK4srXtmcZO2Im45RAiVXyfpBAmmiF3eZWCbKe7qBC4rDRh
LZG4RQW/S86Da0BID7gQz9IFSkaG504MsDhvnA7iAqaHUHUepCsiwsF8BBgBCgAm
AhsMFiEE/rkgny8vP0ZkhB5VrBB7OGaS2t0FAmUkSiQFCQ+P/Z8ACgkQrBB7OGaS
2t3AwA/9GkXKUgvjKGCxwE4SdDt7c2jw6to2TTP9iFJ3Xbk3+5BURT3gkZCuu9D7
gt+97aVo/B4EM7Xz8DQKyY7Ic9VAwDRra/Hwi1V0hw1zyIWQ/gAnX3baU6qLRWHR
vVR5meV8r35C+rg9DaWFYmvS7PIv9LfxESwBPUjbmx8k4/5EJpHUwf12bzkTnot5
7q5lHxKQa6IvqQak+Hp9ZM2KPdsgKO2HWJJIIvYcI5byW9zBKV0O7YR8gtRAJKp9
IbtsXx0WT6cqHOFVc5SSzdcaMt0gLFl7BTnJyvKK2l9GABGBmzYDjeCyF2J+Ippf
oqxqfTe6EoOsuEMc2PbLTs9SsWjyCC2VGlX8+uUH9SoKwL0VQ6LFsP6fhkVKqi/a
rB6UuPR/iZnrKIuxMNQ4U+t2Q6UdMlmXsAXTNdkwzoK9oJRokIrH0ZV1KtH4sjjA
tCic+tOddq+GQLiKe2WpJfxlA0uESCB0TxjAwQmfn1H+dUhPeLlbNimHlH0/hXPd
ifuNGozzADIRseQDyzjl8xGL1qRZLD3cfmda6RyZ+S3dQRuaRrcFCDccpY/pO+F8
jbx64zyqqNs+KV+SkQGOcKFhWTZGCfQ/zMDtDmQKjb3eTAkv1zdEOMw9zEjjmS0q
8FNl+2wO3VnvXwvBbtDdVCIaIq+jVcsy5XtnnV+bJ19Q9yue/XvCwWUEGAEKAA8C
GwwFAmEyoZoFCQueVRUACgkQrBB7OGaS2t1uHBAAhOYVvrtchRmzCvdNER1DtkIs
bgQPJ9OxbyfvmvoD06qxH7PrycLZKbt7yYpAUU/CMc86GwaEe0I5Nm1CTs6NvDIv
g3e7EPIS859tyQflbM56NlwbsopCuoCJYknuroIf/M6dW6vJKNXLMmnL/AtalUBw
X+5pblmGUUJep49oTOxQEnvnuqyvaGjXgFXix5PVFJD2ed5NnQeFpvfCpc/ioNOj
z7ORO82j1ht5nWqPraXX5AYhQFM/kwR1cK4LV7gVDd/q+dfGYHzpxQ/HtyX/Lasi
N6I52QqA95SM1ZZLPFLaNh6EvnB7uC9pLCYS8nvilX7/cez5PFff1e1gXCOT0jv3
mJ2exLmXV0BbfKgjccFCxhrdRLtukfiDfJkySy1zdscnpfng8wJ3xKRv43cUTz7M
Z24OYNMqK26aJZVXEQUYjCwsBylY/F5wjYAwgwZ8yF5RFix28P/K8JsIHb3QrAJK
sNWQAb03ZWis3N3spR5M9Mw3VuDZ3WUXq7mxB5M3kpVoZ3vETU5cwTbADYNPf4Sw
BDK2uIVtxabezxSBtz0FcyYoF+OW8q7r4WvoyC9/+3GfnozZLJcEIVDk4W2pMW4A
UhG/6drKTm3HkSDWIDu7d1sHWMffLEYfUHtN5DKkDkGoPfHvZvu9teR5yLfUrPTf
ktihPn/JMrmwa9pwi8LCwWUEGAEKAA8CGwwFAmCavPcFCQsGcHIACgkQrBB7OGaS
2t0uaA//UWRaRiHEAKeRqBG/T2ak+XZJNu7QHfNgoUEAub9Zru8oPPXx2AJLcHEN
```

```
KWmeFlLxADdWOZs4Bm9oOew3VQnR/dBqjnXfob9Rc+eYUjA3rXazM/QrqcU8Syi3
MjNGUmjdL5aQF+IppAMgOBLG1TEnM7C5/PvrGJuYpGEnkKEwMK/GYhqg2V60pHEV
Pvs66mefJpCzbZSy56qtknSt6yBNWc14XgDX6VTn2kW4CV/3vVJUuvjvYs9SPyY8
mKEXa6QvUd3PcXv6RiWk4lGYuT1+jh2VkcFQ+JnUwv9TbKFB9b5jq1bvW9+LMDEl
YXux7pBP5RPk+OLpyiExIRFWhi3x7aMWOzQ+I9yuNTeYkTHiEAQRUhs/1Fh4oLgI
v9QZgC0mRSN3zm8plQdivs1ZlAosAqqkA9BQwqsgosQe7P92irYIJqay0si9wGCD
wSMsmeXdIF6wW3/UMJZl66aarPeiZApGX0QdTZwjMh/QK/8gTKyeZulKmNkNfwWq
O17OirWqLKssVHtg3VUM8EIdh+oNqDDXSeWtYUmpPpWp+yWZ0x1MFFZhUQHQZTGu
TIj4A92LQzbrfj/jXRvWm2SrJMivUoiDUn+qxKIpVwFlI5gVb+uyTFhw89PCkphr
JwRi052RLoU9yd6Ek46UH4XfZZWrZuzY+zzB7oqGONphLgi/h3DCwWUEGAEKAA8C
GwwFAl771b8FCQlniTUACgkQrBB7OGaS2t2/MxAAjoEGPdzavhsOlXdPCRd1D5QJ
r8T/NSEV2z1cp8ZvdrkjNF09TBP4qsBnKJiuvY1Iw7OGX9W2okvXxgJizE45v9MH
WEMz4hmIjmAfRwcqENgpOc1IY/T0/+kkCW8dB6d30J1kT0n2PCRzN9L5vPqZXGTG
mLvd9MOjH1256w4uxLb+e1HMDTCqEN1ppq9G+EAR/29q8JZWs1marbZZWxSWcg/E
1YYbNafzklgjq4CLh/j8AEWSvLr39zRy9uvQ/yqAKZ4K4aZfh/SPupGDvsD6ZK54
EPHxErQ7aiXTbUHtvwhxWLOP6WmxFA3Shr6L6YUb6jq+0PVliFC517g3mxFHJtwt
yXGNIKhmzmr0l9OlsHafulJ/9QPfK3Ce32SkPhW/11MYA8HzduMv5Arp7cBczXSP
EUTmNIVKv3gTjSQrzRhwhHmMuqyDZ/rXQQ1jl2sxIDjO4MUMvVjYKF+OCNm42gVs
8ca3/wN9ZNU6hyFWeKQDuCAqPPbT5GO/DKseFEwB+07wwyH1RXbyl0v4fneg605X
S7lqhNtw2p1hDL0HYHDiV+aPZ+LoOmX6+dmnqE6bQJaIlVb922KWmliO7F3DkqP7
0jFlhoE1gfiXWkxP4Gy8wOobNfEMgvz02djkGQy+oQqeNdIcZFZgzPTGKB/nVgpt
9CcRDWjPltFCd2e1FBbCwWUEGAEKAA8FAld1gAUCGwwFCQeGH4AACgkQrBB7OGaS
2t1PIQ//Qc5VYfBCxpaMysaPQ44wXPEZSjxIGZhhMGzb1UzzAEYOw+RgKN5nNTXq
L2KoOkOrGnKqZOKByMdXwIPH/rGwwEsbbIpopnibf5ic5B/+xCTIK+qLIwX2ZLuk
NhbL6Y+E+7DxMMh+KqBWHONKkgwVY+rFWOfoops839ABKvc9/Ry4/qqkcb40AzpD
l1iQJ5vK/DMuaDWxWeKXqJLIl3WMGPcPfheuBZL1u7LEEHYKMgzvpbF81WIn3MBo
8jvxf2/o+kMafSSDqgvOu6yu8GOhmScpCbRJn7jV/HrG+tM+zy48TN6/MkGWSR7q
TD34pqBjyatVfVl6dGD6xj/i/Emt5hZB6qXruCDH7AWMoNx+FkDubs4sc4PKysZU
Itya6KdQFo2UeYsNwZhdn6QwKhd85um4JUHJCY0mARvjsQgWXH/5MR40cow77bbE
vVq0XNd+QRVlyT42CEtnIUOFLeDVuZrum5Tuvvna6ImMDoi/z6QcNeL79XsY2m6I
QVRiHr1BDb/8JLkfnWiwL8GRv169Kf8unx0y5u1YBpcMYkyDD2+pnnk3TY0rR+8X
8goecaS8fbyu/Q48K85ZMD8wKW/bzLQ+tK9y8xed24u2QERftMhIw9b6f45Nrrf/
PhgV8RnuwUusSbdDe8kw3eYTmLdzD4kZc9K7SdO2CqT+hm//9JI=
=uGHC
-----END PGP PUBLIC KEY BLOCK-----
```

이전 키

> **Important**
>
> 이전 키가 만료되기 전에 새 키가 생성됩니다. 따라서 특정 시점에 여러 키가 유효할 수 있습니다. 키는 아티팩트가 만들어진 날부터 아티팩트에 서명하는 데 사용되므로 키의 유효 기간이 겹치는 경우 가장 최근에 발급된 키를 사용합니다.

만료 날짜: 2025년 10월 4일

키 ID	0xAC107B386692DADD
유형	RSA
크기	4096/4096
생성 완료	2016-06-30
만료 날짜	2025-10-04
사용자 ID	AWS SDK 및 도구 <aws-dr-tools@amazon.com>
키 지문	FEB9 209F 2F2F 3F46 6484 1E55 AC10 7B38 6692 DADD

SDK for Java의 다음 OpenPGP 공개 키를 클립보드에 복사하려면 오른쪽 상단 모서리에 있는 "복사" 아이콘을 선택합니다.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Comment: Hostname:
Version: Hockeypuck 2.2

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
zSxBV1MgU0RLcyBhbmQgVG9vbHMgPGF3cy1kci10b29sc0BhbWF6b24uY29tPsLB
lAQTAQoAPgIbAwULCQgHAwUVCgkICwUWAgMBAAIeAQIXgBYhBP65IJ8vLz9GZIQe
VawQezhmktrdBQJnABcIBQkRa8qDAAoJEKwQezhmktrdl1MQAIwEuDar3OTxkfTa
cPNKDnNzxaWqrxZ3FTQ+PyrHhQ6usxxrvDKJS+uCjE9bmWHVFU1R4yQNF+72lJdw
5UhX0u+ZgT9afApE65uAZuwLhPsz8upXT8C6VeKXh3shdw7qXi2hrwtM1a0Pls4O
Cs2C9rLUDMJTySrVDDVwpnaAB+8DcFrs9bIt5Q3gdOUatdzDvcb7QKh9jUvzCpbE
cInb1epDN5MRzowMR4iU2VV1RzLxCvm7CQSyXfgf0DFLkXWiknhOq9eINmytJFG/
ntFdiZFkNZ5hP709loywdfNrmqB6PsF8BPGFh8gKw1pjowrfHpv6cNIqShmA76LT
3OHViOlqGFB7obffq//eZGPR0oYJFDrOdD2CFRoHnP3N++AfkA4SRN7eXwyoz6Pk
Do9WNIEEkAcp6PGvv7AokogDo/4OqmxgC6fN+3BTOstWpv4FlD4NxOZWsTs49wxg
kPlCCVf8t75aZZkcjXng1eClZZQ5SBlRtSB7gMqtP7MIn2J5w8spNbs5xQvJc76u
NvzwEasPkY+UcHd05RddOUwoKqDerLUG7YqdlNCJoQR1mBIgZButbQlMyaZcmQqO
iR0kwDi9h6Dl6fnUb2dFCNJw+eDHvsjG8HI3IVZMl0bUQ2kmmwr1O2YQ1ynJQm0l
lMllI4hFU8/lHNHm8ie5darpVXQEwsGUBBMBCgA+AhsDBQsJCAcDBRUKCQgLBRYC
AwEAAh4BAheAFiEE/rkgny8vP0ZkhB5VrBB7OGaS2t0FAmUkSiIFCQ+P/Z0ACgkQ
rBB7OGaS2t3HVg//S+/KbeOBf+nCdHsrWtp9kxvWIpAGvQhIbxx1tp/impfm+5Rm
fKPD0KX+g42fuMmOdDE4gjO4GjGd7ZY3bx+0zbDSdVebzmYCbPZ/BDP990oPKidd
w6Gl8PaIyqfuARKOESBETvAwNgwO4t2ocjs4pYZV+CuHvESYpqkuHjmHtye6ajZW
Mv24NhjVo4EfP33dPugTjXLjeuGT7qQpsYV3a66juHmPVkXwuPqxh9wTNc5TU6FG
UPSfIGMPL0xha7Rg2i5zvRaAxx4bHqGO8IAz/l/E/tJkV5xnt494HQam9UDbiFI0
Q0TSve1R6S45/UjQW6cycyduHtk72s9ipa9YM0ilTdLgKMWFjzYv4h4qeYvLw3oB
JGQew+I0I4dIrwL/TKet33EuFfwmyT9MaJBhqV6geFaQ0uVmwvzpAcvxIoSqSpkJ
B/kASqCEM/o0QZLuWc56cDsmMisD0ouVPt+c1Zk7AWLlf6j8LKYTbK0QxLRh/eeZ
jhSf3HnpaCfonbOoHmeo5d/o3EZ0AiA4GbT3xgScoIgXOT7KGgOWmtWkdyd3Zvl/
o6q6Hwpg4RsqCkwnfNm5ZIvmTAYXWc2hiICSicxrP0fek5Cc4xVgJR5RMNGyI7+m
ut1SE2WvLhMCwEyl5ecWF0tUze8VBlWkHJp0Y4k2ado39Zq/DZrTRQYEVrjCwX0E
EwEKACcCGwMFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AFAmEyoZoFCQueVRUACgkQ
rBB7OGaS2t3axA//dgcz5T4Fs7LWdIQB/KRnvX64IzaGQcwt3FBhYH+sFaSaD+lu
752vi3jlGxMBKs2NFxk6e2U8xBEir3vfKYd+mZ5L6egXC9MYfvMO/UFEFH3A+t3V
dTOJK4RQcaL9+rFRVdDmZuifN9OFfy25d66JCZ5OiqgHTQViVmbrGw2cQdyNWXrq
YLgg+gktadkmzis4J6hF/le8NOBfrG3n+QthFl/v2ppYYW9pmmxzUIf6tAlR1Vr8
PhPukjuFfrPLRL3XPiK4LkdlSI5MX7Llq4RkcZN1NY1LWS8699wJOLRcr8aQYvzZ
Jm7tfZUaekJ5ScXJWJEaWT4poMbWxXINj6VwE+DqKWvjkzoxBXSIdLk4XThA1dIq
3n5SfMkfuWV2A2xHqJtEzI1XeTm/d/JRxIG8hjIs5FNMGJUSNANJuTVA2putCVf0
JbP4QlafXoEVOYwW/EFJY+brjQadwc/knf/QxszDcKb3THuTxR80A0h6ZysmtLEg
PCaD1xUCDdr4P7DGOtV7yMaDR6O8QKmb7TKzCKCPnHouqPT0DhSB2MRq437+mfSe
EGga/sPMxe0z8ugO2CnrCf3eplU3ZOy4eeQSThTKe0Hp5YlsF1cdZSvjt7GJaHR2
9A22nAJY9PojtlM+0Dkr/PH6r2brv3sEuACRNhzqCWhAve+zVnVLeb+Fk1rCwX0E
```

```
EwEKACcCGwMFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AFAmCavPYFCQsGcHEACgkQ
rBB7OGaS2t3fqg//asHYSTBI4IoLibSF5ZJ8NaLo4EqgRtsO0W1zr8fCoFbxI+yI
qWrNXR7eoFj8tWO7Tj6S0kQr4QZcucLeILfox6CtDzO3f3WQTH9m/0si5U4Jf3RA
gBd0vwxVSSSNEsIUUfylOBHLVwlhaTVfh1h1dZhzb18LA1Vqu0100GGxvaG3dUl4
oMSRSK8EeaSO4hIM8ScjVyhjsuPRm1jlwd1EXZ5mkHRGRMGmwi3XysJjIeQRFmuG
a9uPes7I/K85r7X91BLz+G/mkSZsrHxzLxF0mSZtQdhq08GyMeQlJavs7sb1UVbg
ag30JEAjgqRsNIQOMIvl2/amrRJ7DUyQu5YkZQsAyCIhSVZw6zlJlkYVKSw1Cu1I
VX1n/Aahx5wwaQZA1dDTolX8WvL9O/KmEGWbKUSov40uoRwSOeXP3QhW75kD4zT0
n3pSUc7tXka8GAz5Ar+r90l4wc9as2WjWADo3CX8cF0zQOrN1naMQ++xBIAG9ms8
y3L4ECoRqvjCpjUAfhflxwSM7uclCgFBINFKSLV+QGxzWfHjg3+bSQd0hrRn9j4z
Di9aJmFESQ6iykbUjiUFrcI1NUCFKz2awaxh+Sq8UkYcvuxljxdK/zYYEG8DSdQ2
uwlssHCjYVpAb16hq1EAaSqnhv86020G4Z6JCg3AUZt9RlMBpK2Pn/NmPSzCwX0E
EwEKACcCGwMFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AFAl771vAFCQlnimUACgkQ
rBB7OGaS2t35Jw/9GhpQquJVUCAsc4az7+ad8q2d9OUKXOLl2x0j3/ofS8umZJYr
8KXZxPqBqvLjlUyAB5Ur4fWVBdp9iP4Vj74jmRih7YatK8heGmNoUAnI1/90qV5O
ypFObfvWLxvcUNizmS/LYKdNuYcHwyw4bFyTz9gd3XH6Dxak7YiAXz3JgJIXcIB3
ELfpsduzpncLeVwz/dKhYX5iJ7Y/xd4Ew8Ian8FyNDNRwcXobTpPAMNiGLG5xSLq
8RgQfkm07BVVDtu5tPw4V46gnBXGXNjPhSirGMoNbKZqtPO7TcBQhPQ9c95x73hs
kAqZKHrwi1B7cfy1I8y5sRFbPa0RXeVWQKEMZdwLsFhCbEex6iXHP+rMK0nSJf0G
9lF2eJk140n8K7wzakOnjvN00VfN/9D+xD2lCXCzbYbaDXX2tYOd4GS72fW9M/zT
96tIH5UtMGMOICuUJjnL34EbzBuxwTEnJKhDGQH2P+eUzM9jmuCTRLLGw2P0Zuof
6GsSNLf+5pw3BIvEZw5PYT1N6i3ml9MQ7p9BWr7flCF8CU/xzyPN7TVb/677Be7V
SL1rNFLNi0IdqcbLJ63pTWaUGkCSqRnMfq3cIDlzNz2LoVvOO8VVmdtFRkhHN8hJ
h7CUXlaqBOfaJhV3FqA9G8sXmSN3r3pusZb13kfCKsJUZorThWxdaUBuUH3CwX0E
EwEKACcFAld1gAUCGwMFCQeGH4AFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AACgkQ
rBB7OGaS2t1aKA//dBaXto7zUFRbJvLgcSE3hJ0ChYZj8Uuo7iYK26mVycyBFQJK
Iv2XYFIJMwK1Rx1Ja1jA6BIKE3aYyx2LVTYU1Hke826dhH+kV2qN9C6t/VmYpV4b
n/i64po7EzD17ykrm5gB+z47uCvyC79/r8Ouns1mTf/JUql/hYJj2hf1MDWr07/X
Wc1zBLjlT93tg/43Vgz1wFdrUOcNx1+bQGXKTOi4AXSp3UvCL+2YsQ2/JsPf7ZzZ
awSuUVkZJrllp87S1MhZeHmTrCHkV1FHJyudLJErcH0337Jpd8xDRek7Klrx2pAS
cKIiyeAMIk/G1OuExYqEEVFQzMr9ZlMNuhNBJAMr5hVGsTgrwy2hlIrBktXav07d
OleSlsqw9ViZ7F6t90x5l+NkwXVsLsGYSzuNyIfomNuoCgA+cfM3TjzVp41qsg1J
WJc9Bavaan2pKF6Lb9Fq8u3HZk2u+YZbvZkqkXwTdZZQOkEmoVV4Y1G86bdpmPyj
8eV7C02NxPii4l+qV8qJQu/6DsA0QwMtBMUNODm3BF2+ZmUHuhMGxq9/4vDE8heE
ffYhHtNftV6JwwzGZmeZkrYA1P9AGLeVp/6iNUe8H5/oPvh4s2rRmqN+L/dQU1ix
iOAT5iAKoRQGkduXrWc4fAY5KxDB9qna4oqXO6QP8rEflI8ELcNgYEj4oJvOwU0E
V3WABQEQALzM0Cs9Zvd08xOEvbEBj59LrS9d0HVKQ61gmkNakWC+jR35VD6FXpe6
UYAcBLrEbVYfKw9P0p6MhFKAsb570JoznKGzE1rVYUZQzhD0RKje35rvkajvEcjG
AWMLTjr87pWHeD0389ER64bzORncfa/l+YP56PI+CThb2wUvTTONGJkPQUpVhH+P
256cQL/Y0Fwu4XLerpwN+YKgMQ47raRcydobPeSfMQr9fVKRyOzFEOrvNpCVDUqi
77d0gLDLjHlIlDyOX5554S8XYLb91eYOiFvnu2pTCKiiExRCSYK29mAQePKlTCCn
QxOjbmBbGS8mVIkpQ5vpvXvzpY3JIjMXaDGqWSQSYGXhECyxCR5eOtKYbCwwPIc2
rIl5gW6yXyw9pKmj5XafTP7YHTvRSr7CZ/VLkDkWl6AfQ9nPOg1mjwjpDFpmN71h
JlSKMaZkh0QGV5FW3dK+GLwxiWdqx3htbZErWyvumWQF/xBF7puKJBEXcoM5KfkJ
```

```
uZekBwcnVkfNFF2RdkM1ALq8InGzLXc7ROuEm0BXVirfju7JRtWLb3UhJWCuhRW2
muyYegSTkag5MduD1IJK37GL8WIlAL65taYgZegUoxHdSaEOefOhspxuduz8d33z
UV1WCFhi+r/+BMCQmTRbF8ao7fTC1dGd084DRP6qE/dMT4u0ZEn7ABEBAAHCwXwE
GAEKACYCGwwWIQT+uSCfLy8/RmSEHlWsEHs4ZpLa3QUCZwAXCwUJEWvKhQAKCRCs
EHs4ZpLa3XtzD/9dwi1qffV70UTq8w/21jn1owHp09jxP7WHTmPWHE0BW5yFIWlV
A1gKN6Ym0dw+LvS5WOKJaRnyewUyBxWvZsn6Wlb5qzY7nmCOKJpYtuCUPwiqjXWP
EM8c/v0MojSuwMOXBAViLvOFhgdUrHn1lk962XvWAW++4DXFh2deaV0163IFMRmO
PNPDAiPWBVqvBANIh2sLRZ5gd1BXwpVrd+x8tzyr69YrN7hutPlCyPEUM9//mcEh
vFPsbW/iOx/foCE3NXhQm/rSMKecVn5csXBV2JOlMzi+8txYNrSBLkjbSB1AvTQ1
aG3+nCNCgM2XDLyoj0IrgZ1To4Ay5gmTOR+msY/cfoIuKFYenmtxy6jM8o5uSZHg
hoClrx9IA98hhGQ73G2r5EDpXuU/uCXn53Sswj65bl9IssfqEIoji/FonkkpEgeg
bGXFDUnrhicDO/WOzqpXf2Fa0DQWY+Vc/pt52ftBFgwzCNIUYDKUhCHPnZ0wtLtd
N2fkXHNiCavCDZlOud7FHHwmRNdj2q1uKxe4m+pFYmKwAU/H+Htkz9Gjsj+ZKedY
nnfai2s2gQOrbfwvV9VdhCWSuLK17ZnGTtiJuOUQIlV8n6QQJpohd3mVgmynu6gQ
uKw0YS2RuEUFv0vOg2tASA+4EM/SBUpGhudODLA4b5wO4gKmh1B1HqQrIsLBfAQY
AQoAJgIbDBYhBP65IJ8vLz9GZIQeVawQezhmktrdBQJlJEokBQkPj/2fAAoJEKwQ
ezhmktrdwMAP/RpFylIL4yhgscBOEnQ7e3No8OraNk0z/YhSd125N/uQVEU94JGQ
rrvQ+4Lfve2laPweBDO18/A0CsmOyHPVQMA0a2vx8ItVdIcNc8iFkP4AJ1922lOq
i0Vh0b1UeZnlfK9+Qvq4PQ2lhWJr0uzyL/S38REsAT1I25sfJOP+RCaR1MH9dm85
E56Lee6uZR8SkGuiL6kGpPh6fWTNij3bICjth1iSSCL2HCOW8lvcwSldDu2EfILU
QCSqfSG7bF8dFk+nKhzhVXOUks3XGjLdICxZewU5ycryitpfRgARgZs2A43gshdi
fiKaX6Ksan03uhKDrLhDHNj2y07PUrFo8ggtlRpV/PrlB/UqCsC9FUOixbD+n4ZF
Sqov2qwelLj0f4mZ6yiLsTDUOFPrdkOlHTJZl7AF0zXZMM6CvaCUaJCKx9GVdSrR
+LI4wLQonPrTnXavhkC4intlqSX8ZQNLhEggdE8YwMEJn59R/nVIT3i5WzYph5R9
P4Vz3Yn7jRqM8wAyEbHkA8s45fMRi9akWSw93H5nWukcmfkt3UEbmka3BQg3HKWP
6TvhfI28euM8qqjbPilfkpEBjnChYVk2Rgn0P8zA7Q5kCo293kwJL9c3RDjMPcxI
45ktKvBTZftsDt1Z718LwW7Q3VQiGiKvo1XLMuV7Z51fmydfUPcrnv17wsFlBBgB
CgAPAhsMBQJhMqGaBQkLnlUVAAoJEKwQezhmktrdbhwQAITmFb67XIUZswr3TREd
Q7ZCLG4EDyfTsW8n75r6A9OqsR+z68nC2Sm7e8mKQFFPwjHPOhsGhHtCOTZtQk7O
jbwyL4N3uxDyEvOfbckH5WzOejZcG7KKQrqAiWJJ7q6CH/zOnVurySjVyzJpy/wL
WpVAcF/uaW5ZhlFCXqePaEzsUBJ757qsr2ho14BV4seT1RSQ9nneTZ0Hhab3wqXP
4qDTo8+zkTvNo9YbeZ1qj62l1+QGIUBTP5MEdXCuC1e4FQ3f6vnXxmB86cUPx7cl
/y2rIjeiOdkKgPeUjNWWSzxS2jYehL5we7gvaSwmEvJ74pV+/3Hs+TxX39XtYFwj
k9I795idnsS5l1dAW3yoI3HBQsYa3US7bpH4g3yZMkstc3bHJ6X54PMCd8Skb+N3
FE8+zGduDmDTKitumiWVVxEFGIwsLAcpWPxecI2AMIMGfMheURYsdvD/yvCbCB29
0KwCSrDVkAG9N2VorNzd7KUeTPTMN1bg2d1lF6u5sQeTN5KVaGd7xE1OXME2wA2D
T3+EsAQytriFbcWm3s8Ugbc9BXMmKBfjlvKu6+Fr6Mgvf/txn56M2SyXBCFQ5OFt
qTFuAFIRv+nayk5tx5Eg1iA7u3dbB1jH3yxGH1B7TeQypA5BqD3x72b7vbXkeci3
1Kz035LYoT5/yTK5sGvacIvCwsFlBBgBCgAPAhsMBQJgmrz3BQkLBnByAAoJEKwQ
ezhmktrdLmgP/1FkWkYhxACnkagRv09mpPl2STbu0B3zYKFBALm/Wa7vKDz18dgC
S3BxDSlpnhZS8QA3VjmbOAZvaDnsN1UJ0f3Qao5136G/UXPnmFIwN612szP0K6nF
PEsotzIzRlJo3S+WkBfiKaQDIDgSxtUxJzOwufz76xibmKRhJ5ChMDCvxmIaoNle
tKRxFT77OupnnyaQs22UsueqrZJ0resgTVnNeF4A1+lU59pFuAlf971SVLr472LP
Uj8mPJihF2ukL1Hdz3F7+kYlpOJRmLk9fo4dlZHBUPiZ1ML/U2yhQfW+Y6tW71vf
```

```
izAxJWF7se6QT+UT5Pji6cohMSERVoYt8e2jFjs0PiPcrjU3mJEx4hAEEVIbP9RY
eKC4CL/UGYAtJkUjd85vKZUHYr7NWZQKLAKqpAPQUMKrIKLEHuz/doq2CCamstLI
vcBgg8EjLJnl3SBesFt/1DCWZeummqz3omQKRl9EHU2cIzIf0Cv/IEysnmbpSpjZ
DX8Fqjtezoq1qiyrLFR7YN1VDPBCHYfqDagw10nlrWFJqT6VqfslmdMdTBRWYVEB
0GUxrkyI+APdi0M2634/410b1ptkqyTIr1KIg1J/qsSiKVcBZSOYFW/rskxYcPPT
wpKYaycEYtOdkS6FPcnehJOOlB+F32WVq2bs2Ps8we6KhjjaYS4Iv4dwwsFlBBgB
CgAPAhsMBQJe+9W/BQkJZ4k1AAoJEKwQezhmktrdvzMQAI6BBj3c2r4bDpV3TwkX
dQ+UCa/E/zUhFds9XKfGb3a5IzRdPUwT+KrAZyiYrr2NSMOzhl/VtqJL18YCYsxO
Ob/TB1hDM+IZiI5gH0cHKhDYKTnNSGP09P/pJAlvHQend9CdZE9J9jwkczfS+bz6
mVxkxpi73fTDox9duesOLsS2/ntRzA0wqhDdaaavRvhAEf9vavCWVrNZmq22WVsU
lnIPxNWGGzWn85JYI6uAi4f4/ABFkry69/c0cvbr0P8qgCmeCuGmX4f0j7qRg77A
+mSueBDx8RK0O2ol021B7b8IcVizj+lpsRQN0oa+i+mFG+o6vtD1ZYhQude4N5sR
RybcLclxjSCoZs5q9JfTpbB2n7pSf/UD3ytwnt9kpD4Vv9dTGAPB83bjL+QK6e3A
XM10jxFE5jSFSr94E40kK80YcIR5jLqsg2f610ENY5drMSA4zuDFDL1Y2ChfjgjZ
uNoFbPHGt/8DfWTVOochVnikA7ggKjz20+RjvwyrHhRMAftO8MMh9UV28pdL+H53
oOtOV0u5aoTbcNqdYQy9B2Bw4lfmj2fi6Dpl+vnZp6hOm0CWiJVW/dtilppYjuxd
w5Kj+9IxZYaBNYH4l1pMT+BsvMDqGzXxDIL89NnY5BkMvqEKnjXSHGRWYMz0xigf
51YKbfQnEQ1oz5bRQndntRQWwsFlBBgBCgAPBQJXdYAFAhsMBQkHhh+AAAoJEKwQ
ezhmktrdTyEP/0HOVWHwQsaWjMrGj0OOMFzxGUo8SBmYYTBs29VM8wBGDsPkYCje
ZzU16i9iqDpDqxpyqmTigcjHV8CDx/6xsMBLG2yKaKZ4m3+YnOQf/sQkyCvqiyMF
9mS7pDYWy+mPhPuw8TDIfiqgVhzjSpIMFWPqxVjn6KKbPN/QASr3Pf0cuP6qpHG+
NAM6Q5dYkCebyvwzLmg1sVnil6iSyJd1jBj3D34XrgWS9buyxBB2CjIM76WxfNVi
J9zAaPI78X9v6PpDGn0kg6oLzrusrvBjoZknKQm0SZ+41fx6xvrTPs8uPEzevzJB
lkke6kw9+KagY8mrVX1ZenRg+sY/4vxJreYWQeql67ggx+wFjKDcfhZA7m7OLHOD
ysrGVCLcmuinUBaNlHmLDcGYXZ+kMCoXfObpuCVByQmNJgEb47EIFlx/+TEeNHKM
O+22xL1atFzXfkEVZck+NghLZyFDhS3g1bma7puU7r752uiJjA6Iv8+kHDXi+/V7
GNpuiEFUYh69QQ2//CS5H51osC/Bkb9evSn/Lp8dMubtWAaXDGJMgw9vqZ55N02N
K0fvF/IKHnGkvH28rv0OPCvOWTA/MClv28y0PrSvcvMXnduLtkBEX7TISMPW+n+O
Ta63/z4YFfEZ7sFLrEm3Q3vJMN3mE5i3cw+JGXPSu0nTtgqk/oZv//SS
=bboB
-----END PGP PUBLIC KEY BLOCK-----
```

만료 날짜: 2024년 10월 8일

키 ID	0xAC107B386692DADD
유형	RSA
크기	4096/4096
생성 완료	2016-06-30

만료 날짜	2024-10-08
사용자 ID	AWS SDK 및 도구 <aws-dr-tools@amazon.com>
키 지문	FEB9 209F 2F2F 3F46 6484 1E55 AC10 7B38 6692 DADD

SDK for Java의 다음 OpenPGP 공개 키를 클립보드에 복사하려면 오른쪽 상단 모서리에 있는 "복사" 아이콘을 선택합니다.

```
-----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-----
```

문서 기록

이 주제에서는 지금까지 이루어진 AWS SDK for Java 개발자 가이드의 주요 변경 사항을 설명합니다.

변경 사항	설명	날짜
the section called “현재 키”	2026년 9월 27일에 만료되는 새 PGP 키를 추가합니다.	2025년 10월 1일
지원되지 않는 코드 패턴	마이그레이션 도구의 지원되지 않는 코드 패턴을 문서화합니다.	2025년 9월 23일
마이그레이션 도구	GA 릴리스에 대해 업데이트합니다.	2025년 9월 19일
지표 참조	SDK에서 수집한 지표에 대한 세부 정보를 추가합니다.	2025년 9월 3일
the section called “`VersionedRecordExtension` 을 사용한 낙관적 잠금 구현”	DynamoDB 향상된 클라이언트의 `VersionRecordExtension` 에 대한 세부 정보를 추가합니다.	2025년 9월 2일
1.x 클라이언트를 사용하여 애플리케이션 찾기	버전 2로 마이그레이션하기 전에 AWS CloudTrail 이벤트를 쿼리하여 AWS SDK for Java 1.x 클라이언트를 사용해 애플리케이션을 식별하는 데 도움이 되는 설명을 추가합니다.	2025년 8월 20일
Transfer Manager	클라이언트 생성자, 메서드, 모델 객체 및 동작 변경에 대한 포괄적인 마이그레이션 테이블을 추가합니다. 수동 마이그레이션이 필요한 지원되지 않는 메	2025년 7월 1일

변경 사항	설명	날짜
	서드에 대한 자세한 코드 예제가 포함되어 있습니다.	
지표	`LoggingMetricPublisher` 에 대한 포괄적인 설명서를 추가합니다. 개선된 시작 가이드로 지표 주제를 재구성했습니다.	2025년 6월 20일
the section called “높은 수준의 변경 사항”	마이그레이션 가이드에서 `DynamoDBMapper` (v1)와 DynamoDB 향상된 클라이언트 (v2) 간의 비어 있는 문자열 처리 차이에 대한 정보를 추가합니다.	2025년 6월 18일
목차 재구성	가이드의 다른 섹션에서 통합된 ???장을 추가합니다.	2025년 6월 16일
???	MD5 체크섬이 필요한 시스템과의 이전 버전 호환성을 위해 `LegacyMd5Plugin` 을 사용하는 방법에 대한 정보를 추가합니다.	2025년 5월 19일
the section called “S3 클라이언트 차이”	AWS SDK for Java의 v1과 v2 간의 S3 클라이언트 차이를 추가하고 마이그레이션 도구가 V1 코드를 자동으로 마이그레이션할 수 없는 경우 마이그레이션 예제를 제공합니다.	2025년 4월 24일
역직렬화 차이점	SDK for Java의 v1과 v2 간에 역직렬화 차이를 추가합니다.	2025년 4월 10일

변경 사항	설명	날짜
the section called “SQS 자동 요청 배치 처리”	SDK for Java의 v1에서 v2로 SQS 자동 요청 배치 처리를 위한 마이그레이션 콘텐츠를 추가합니다.	2025년 4월 8일
???	자동 체크섬 계산에 대한 정보 업데이트	2025년 4월 3일
the section called “ALPN 구성”	Netty 기반 HTTP 클라이언트와의 ALPN 프로토콜 협상 구성을 보여줍니다.	2025년 2월 21일
the section called “Lambda 함수”	AWS Lambda에서 EMF 로깅 지표 게시자를 사용하여 SDK 지표를 캡처하는 방법에 대한 정보를 추가합니다.	2025년 2월 6일
`ContentStreamProvider` 구현	`ContentStreamProvider` 를 구현하는 방법에 대한 주제를 추가합니다.	2025년 1월 29일
the section called “체크섬을 통한 데이터 무결성 보호”	자동 체크섬 계산에 대한 세부 정보로 콘텐츠가 업데이트되었습니다.	2025년 1월 16일
the section called “Amazon S3 작업”	Amazon S3 작업에 대한 마이그레이션 콘텐츠를 추가합니다.	2025년 1월 8일
the section called “AWS CRT 기반 HTTP 클라이언트에 액세스”	AWS CRT 기반 구성 요소와 함께 플랫폼별 jar를 사용하는 방법에 대한 정보를 추가합니다.	2024년 11월 14일
the section called “인증 시 IAM Roles Anywhere 사용”	인증 시 IAM Roles Anywhere를 사용하는 방법에 대한 정보를 추가합니다.	2024년 11월 6일

변경 사항	설명	날짜
the section called “자격 증명 구성 캐싱 예제”	`asyncCredentialUpdateEnabled` 속성을 사용하여 자격 증명 공급자를 구성하는 예제를 추가합니다.	2024년 11월 4일
the section called “자동 요청 배치 처리 사용”	Amazon SQS용 자동 요청 배치 API를 문서화하는 새 주제를 추가합니다.	2024년 10월 23일
OpenPGP 키	현재 OpenPGP 키 정보를 업데이트합니다.	2024년 10월 10일
the section called “표현식에서 복잡한 유형 사용”‘and’the section called “복잡한 유형이 포함된 항목 업데이트”	표현식 및 업데이트에서 복잡한 유형으로 작업하는 방법에 대한 콘텐츠를 추가합니다.	2024년 10월 10일
Amazon S3 버킷 이름 업데이트	Amazon S3 버킷 이름을 업데이트합니다.	2024년 9월 30일
the section called “계정 기반 엔드포인트로 성능 최적화”	DynamoDB의 AWS 계정 기반 엔드포인트에 대한 정보를 추가합니다.	2024년 9월 24일
the section called “bean, 맵, 목록 및 세트로 구성된 속성으로 작업”	복잡한 유형의 속성 작업에 대해 설명하는 DynamoDB 향상된 클라이언트의 섹션을 업데이트합니다.	2024년 9월 6일
the section called “바로가기 검색이 가능하도록 서비스 클라이언트를 구성”	Lambda SnapStart for Java를 사용할 경우 `EnvironmentVariableCredentialsProvider` 사용을 명확히 합니다.	2024년 8월 19일

변경 사항	설명	날짜
the section called “병렬 전송 지원 구성”	병렬 전송 지원을 활성화하는 방법에 대한 정보가 포함된 페이지 추가	2024년 8월 15일
the section called “AutoGeneratedUuidExtension을 사용하여 UUID 생성”	DynamoDB 향상된 클라이언트 `AutoGeneratedUuidExtension` 에 대한 정보 추가	2024년 8월 14일
???	마이그레이션 도구에 대한 섹션 추가(미리 보기 릴리스)	2024년 8월 9일
the section called “S3 이벤트 알림”	S3 이벤트 알림 API로 작업하는 방법을 설명하는 섹션 추가	2024년 7월 21일
the section called “DynamoDB 작업”	DynamoDB 매핑/문서 API에 대한 v1-v2 마이그레이션 정보 추가	2024년 7월 21일
the section called “S3 이벤트 알림”	S3 이벤트 알림 API에 대한 v1-b2 마이그레이션 정보 추가	2024년 7월 21일
the section called “Retries”	재시도 전략 주제 추가	2024년 6월 18일
JVM TTL을 설정하는 방법	Java 명령줄 시스템 속성을 사용하여 `networkaddress.cache.ttl` 보안 속성을 설정하는 설명을 제거합니다.	2024년 5월 21일
the section called “AWS Lambda를 위한 SDK 시작 시간 단축”	AWS Lambda의 시작 시간 단축하는 HTTP 클라이언트 권장 사항을 업데이트	2024년 5월 14일
the section called “지표 참조”	지표 테이블 항목 재구성	2024년 5월 1일
the section called “문제 해결”	문제 해결 주제를 추가합니다.	2024년 4월 26일

변경 사항	설명	날짜
the section called “각 요청에서 수집된 지표”	SDK에서 보고하는 새 지표를 추가합니다.	2024년 4월 26일
the section called “DNS 이름 조회를 위한 JVM TTL 설정”	권장 DNS 조회 TTL을 5초로 변경합니다.	2024년 4월 23일
the section called “artifactId 매핑에 대한 패키지 이름”	Maven artifactId 매핑 주제에 패키지 이름을 추가합니다.	2024년 4월 17일
the section called “지표”	지표 섹션에 구성 세부 정보를 추가합니다.	2024년 4월 12일
the section called “IAM 정책 빌더 API”	IAM 정책 빌더 API 마이그레이션 정보를 추가합니다.	2024년 4월 11일
???	HTTP 프록시 정보를 업데이트합니다.	2024년 4월 3일
the section called “안전”	IMDSv1을 사용 해제하는 설명을 추가합니다.	2024년 3월 14일
the section called “단계별 설명”	단계별 마이그레이션 설명을 추가합니다.	2024년 3월 8일
버전 2로 마이그레이션	마이그레이션 주제를 업데이트합니다.	2024년 2월 14일
the section called “AWS CRT 기반 HTTP 클라이언트 구성”	동기식 AWS CRT 기반 HTTP 클라이언트에 대한 정보를 추가합니다.	2024년 1월 5일
the section called “Amazon Cognito 자격 증명”‘and’the section called “Amazon Cognito 자격 증명 공급자”	Amazon Cognito 예제가 코드 예제 섹션으로 이동했습니다.	2023년 12월 28일

변경 사항	설명	날짜
OpenPGP 키	현재 OpenPGP 키를 입력합니다.	2023년 12월 6일
the section called “직렬화 차이점”	SDK for Java v1과 v2의 직렬화 차이점에 대해 설명합니다.	2023년 12월 5일
the section called “Transfer Manager”	S3 Transfer Manager에서 버전 1에서 버전 2로의 변경 사항을 자세히 설명하는 단원을 추가하세요.	2023년 11월 13일
the section called “주석 참조”	DynamoDB 향상된 클라이언트와 함께 사용할 수 있는 데이터 클래스 주석 목록을 추가합니다.	2023년 10월 30일
???	Java용 SDK v1.x에서 v2.x로 라이브러리 및 유틸리티의 마이그레이션 상태에 대한 정보 추가	2023년 10월 17일
???	Gradle 설정 주제 업데이트	2023년 10월 17일
the section called “중첩된 개체의 null 속성 무시”	DynamoDB 향상된 클라이언트 `@DynamoDbIgnoreNulls` 주석에 대한 정보를 추가합니다.	2023년 9월 22일
the section called “교차 리전 액세스”	Amazon S3 버킷에 대한 교차 리전 액세스 정보를 추가합니다.	2023년 8월 31일
the section called “빈 객체를 보존”	`@DynamoDbPreserveEmptyObject` 주석을 설명하는 단원을 추가하세요.	2023년 8월 25일

변경 사항	설명	날짜
???	서비스 클라이언트 단원 업데이트.	2023년 8월 15일
the section called “클라이언트 권장 사항”	버전 0.23부터 AWS CRT는 Alpine Linux 같은 musl 기반 OS를 지원합니다. 이제 HTTP 클라이언트 권장 사항에 musl 지원이 반영됩니다.	2023년 8월 11일
the section called “IAM 정책 생성”	IAM 정책 빌더 API 단원 추가	2023년 7월 31일
the section called “시작하기”	DynamoDB 향상된 클라이언트 항목의 시작하기 단원에 있는 몇 가지 코드 조각을 수정하세요.	2023년 7월 24일
the section called “HTTP 프록시 구성”	각 HTTP 클라이언트에 대한 HTTP 프록시 지원 정보와 예제를 추가하세요.	2023년 6월 2일
목차 재구성	코드 예제 단원 및 AWS 서비스 호출를 최상위 목차 항목으로 승격하세요.	2023년 5월 24일
the section called “로깅 종속성 추가”	로깅 단원에 Gradle 종속성을 표시하세요.	2023년 5월 23일
the section called “페이지 매김”	페이지 매김 항목 업데이트.	2023년 5월 18일
the section called “Gradle 프로젝트 설정하기”	Gradle 프로젝트 설정을 업데이트하세요.	2023년 5월 3일
DynamoDB 향상된 클라이언트 API	재작성된 DynamoDB 향상된 클라이언트 API 주제가 출시되었습니다.	2023년 4월 28일

변경 사항	설명	날짜
시작하기 자습서 지침 업데이트	자격 증명 제공자에 대한 옵션을 포함하도록 Maven 아키타입이 수정되었습니다. 이에 따라 지침이 수정되었습니다.	2023년 4월 11일
the section called “클라이언트 권장 사항”	HTTP 클라이언트 의사 결정 지침 추가	2023년 3월 30일
IAM 모범 사례 업데이트	IAM 모범 사례에 따라 가이드가 업데이트되었습니다. 자세한 내용은 IAM의 보안 모범 사례를 참조하십시오.	2023년 3월 14일
the section called “프로필 자격 증명 다시 로드”	프로필 자격 증명을 다시 로드하는 방법에 대한 단원을 추가하세요.	2023년 2월 9일
the section called “AWS CRT 기반 HTTP 클라이언트 구성”	GA 릴리스를 위한 업데이트 주제.	2023년 2월 8일
the section called “Amazon EC2 인스턴스 메타데이터 작업”	Amazon S3 인스턴스 메타데이터 서비스용 Java SDK 클라이언트에 대한 안내 예제를 추가합니다.	2023년 2월 1일
the section called “고성능 S3 클라이언트 사용”	AWS CRT 기반 S3 클라이언트 단원을 추가합니다.	2022년 12월 19일
the section called “파일 및 디렉터리 전송”	GA 릴리스에 대한 Amazon S3 Transfer Manager 예제를 업데이트하세요.	2022년 12월 19일
the section called “모범 사례”	모범 사례 단원이 추가되었습니다.	2022년 11월 18일

변경 사항	설명	날짜
the section called “외부 프로세스 사용”	외부 프로세스에서 자격 증명을 로드하는 단원이 추가되었습니다.	2022년 11월 15일
the section called “지표 참조”	HTTP 클라이언트 사용 요구 사항이 포함된 지표 목록이 업데이트되었습니다.	2022년 11월 9일
the section called “파일 및 디렉터리 전송”	예제 코드가 수정되었습니다.	2022년 11월 2일
the section called “AWS Lambda를 위한 SDK 시작 시간 단축”	Lambda 시작 시간을 줄이기 위한 추가 옵션이 포함된 단원이 업데이트되었습니다.	2022년 11월 1일
the section called “HTTP 클라이언트 구성”	SDK의 모든 HTTP 클라이언트를 포괄하는 구성 정보가 추가되었습니다.	2022년 10월 26일
the section called “로깅”	모든 HTTP 클라이언트에 대한 유선 로깅 세부 정보를 포함하도록 로깅 항목을 업데이트했습니다.	2022년 10월 4일
the section called “AWS 데이터베이스 서비스”	AWS 데이터베이스 서비스 및 Java 2.x용 SDK의 개요 단원이 추가되었습니다.	2022년 9월 13일
EC2-Classic 네트워킹 사용 중지	EC2-Classic은 2022년 8월 15일에 사용 중지됩니다.	2022년 7월 28일
the section called “추가 인증 옵션”	Single Sign-On 인증에 필요한 종속성 업데이트하세요.	2022년 7월 18일
the section called “전송 계층 보안(TLS)”	TLS 보안 정보를 업데이트하세요.	2022년 4월 8일

변경 사항	설명	날짜
the section called “추가 인증 옵션”	자격 증명 설정 및 사용에 대한 자세한 정보가 추가되었습니다.	2021년 2월 22일
the section called “GraalVM Native Image 프로젝트 설정”	GraalVM 네이티브 이미지 프로젝트 설정에 대한 새 항목입니다.	2021년 2월 18일
the section called “Waiters”	Waiters가 출시되었으며 새 기능에 대한 주제가 추가되었습니다.	2020년 9월 30일
the section called “지표”	지표가 출시되었으며 새 기능에 대한 주제가 추가되었습니다.	2020년 8월 17일
the section called “Amazon SNS”	Amazon SNS에 대한 주제의 예제가 추가되었습니다.	2020년 5월 30일
the section called “AWS Lambda를 위한 SDK 시작 시간 단축”	AWS Lambda 함수 성능 항목이 추가되었습니다.	2020년 5월 29일
the section called “DNS 이름 조회를 위한 JVM TTL 설정”	JVM TTL DNS 캐싱 항목이 추가되었습니다.	2020년 4월 27일
the section called “Apache Maven 프로젝트 설정”, the section called “Gradle 프로젝트 설정하기”	새로운 Maven과 Gradle 설정 항목.	2020년 4월 21일
the section called “전송 계층 보안(TLS)”	보안 단원에 TLS 1.2 추가.	2020년 3월 19일
the section called “Amazon Kinesis Data Streams 가입”	Kinesis 스트림 예제가 추가되었습니다.	2018년 8월 2일

변경 사항	설명	날짜
the section called “페이지 매김”	자동 페이지 매김 주제 추가.	2018년 5월 4일
???	IAM, Amazon EC2 CloudWatch 및 DynamoDB에 대한 예제 항목이 추가되었습니다.	2017년 12월 29일
the section called “Amazon S3”	Amazon S3용 getobjects 예제가 추가되었습니다.	2017년 8월 7일
the section called “비동기 프로그래밍”	비동기식 관련 주제 추가.	2017년 8월 4일
AWS SDK for Java 2.x의 GA 릴리스	AWS SDK for Java 버전 2(v2) 릴리스.	2017년 6월 28일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.