

版本 2.x 开发人员指南

AWS SDK for Java 2.x



AWS SDK for Java 2.x: 版本 2.x 开发人员指南

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商标和商业外观不得用于任何非 Amazon 的商品或服务，也不得以任何可能引起客户混淆、贬低或诋毁 Amazon 的方式使用。所有非 Amazon 拥有的其他商标均为各自所有者的财产，这些所有者可能附属于 Amazon、与 Amazon 有关联或由 Amazon 赞助，也可能不是如此。

Table of Contents

开发人员指南 – AWS SDK for Java 2.x	1
开始使用 SDK	1
开发移动应用程序	1
SDK 主要版本的维护和支持	1
其他资源	1
为 SDK 做贡献	2
入门教程	3
步骤 1：为本教程进行设置	3
步骤 2：创建项目	3
步骤 3：编写代码	8
步骤 4：构建并运行应用程序	12
成功	13
清理	13
后续步骤	13
设置	14
设置概述	14
设置身份验证	15
设置 SDK 的单点登录访问权限	15
使用登录 AWS CLI	16
安装 Java 和构建工具	16
其他身份验证选项	17
设置 Apache Maven 项目	17
先决条件	17
创建 Maven 项目	17
为 Maven 配置 Java 编译器	18
将开发工具包声明为依赖项	19
设置开发工具包模块的依赖项	20
重新构建项目。	22
设置 Gradle 项目	22
设置 GraalVM 原生映像项目	29
先决条件	29
使用原型创建项目	29
生成原生映像	30
使用 SDK	31

使用服务客户端	31
创建服务客户端	31
默认客户端配置	31
配置服务客户端	32
提出请求	32
处理响应	33
关闭服务客户端	34
处理异常	34
使用 waiter	35
HTTP 客户端	36
超时	36
执行拦截器	37
其他信息	37
向 SDK 提供临时凭证	37
配置对临时凭证的访问权限	37
默认凭证提供程序链	39
使用特定的凭证提供程序或提供程序链	40
使用配置文件	41
从外部流程加载临时凭证	44
在代码中提供临时凭证	47
阅读上的 IAM 角色证书 Amazon EC2	50
使用 AWS 区域	51
显式配置 AWS 区域	51
根据环境确定区域	52
查看服务可用性	53
选择特定端点	53
缩短 SDK 的启动时间 AWS Lambda	54
使用 AWS 基于 CRT 的 HTTP 客户端	54
移除未使用的 HTTP 客户端依赖项	54
配置服务客户端以进行快捷查找	55
在 Lambda 函数处理程序之外初始化 SDK 客户端	56
尽量减少依赖关系注入	57
使用 Maven Archetype 瞄准 AWS Lambda	57
Lambda SnapStart	57
影响启动时间的 2.x 版更改	57
其他 资源	57

HTTP 客户	58
可用客户端	58
客户端建议	59
智能默认值	62
配置基于 Apache 的 HTTP 客户端	64
配置基于 URLConnection 的 HTTP 客户端	69
配置基于 Netty 的 HTTP 客户端	75
配置 AWS 基于 CRT 的 HTTP 客户	81
配置 HTTP 代理	91
异常处理	96
为什么使用未选中的异常？	96
AwsServiceException (和子类)	96
SdkClientException	97
异常和重试行为	98
重试	98
重试策略	98
指定策略	101
自定义策略	103
从 RetryPolicy 迁移到 RetryStrategy	104
日志记录	104
Log4j 2 配置文件	105
添加日志记录依赖项	105
特定于 SDK 的错误消息和警告	106
请求/响应摘要日志记录	106
调试级 SDK 日志	107
启用线路记录	110
为 DNS 名称查找设置 JVM TTL	114
如何设置 JVM TTL	115
最佳做法	115
重复使用 SDK 客户端	116
关闭输入流	116
调整 HTTP 配置	116
对 Netty 使用 OpenSSL	116
配置 API 超时	117
使用指标	118
故障排除	118

连接重置	118
连接超时	119
读取超时	119
连接池超时	119
类路径错误	122
签名错误	123
连接池已关闭	124
使用SDK功能	127
一般功能	127
特定于服务的功能	127
处理分页结果	127
同步分页	128
异步分页	130
轮询资源状态	135
先决条件	136
使用 Waiter	136
配置 Waiter	137
代码示例	138
使用异步编程	138
非流式操作	138
流式操作	141
高级操作	145
使用 HTTP /2	146
使用SDK指标	146
先决条件	146
如何启用指标收集	147
自定义指标发布者	149
指标何时可用？	150
收集哪些信息？	150
我该如何使用这些信息？	151
服务客户端指标	151
与... 一起工作 AWS 服务	155
CloudWatch	155
从 CloudWatch 中获取指标	156
将自定义指标数据发布到 CloudWatch	157
使用 CloudWatch 警报	159

使用 Amazon CloudWatch 活动	163
AWS 数据库服务	167
Amazon DynamoDB	167
Amazon RDS	168
Amazon Redshift	168
Amazon Aurora Serverless v1	169
Amazon DocumentDB	169
DynamoDB	169
使用中的表格 DynamoDB	170
处理 DynamoDB 中的项目	179
将对象映射到 DynamoDB 项目	186
Amazon EC2	296
托管 Amazon EC2 实例	297
使用 AWS 区域、区和可用区	303
在中使用安全组 Amazon EC2	306
使用 Amazon EC2 实例元数据	311
IAM	317
管理 IAM 访问密钥	317
管理 IAM 用户	323
创建 IAM policy	327
使用 IAM 策略	335
使用 IAM 服务器证书	341
Kinesis	346
订阅 Amazon Kinesis Data Streams	346
Lambda	356
调用 Lambda 函数	356
列出 Lambda 函数	358
删除 Lambda 函数	359
Amazon S3	359
使用接入点或多区域接入点	360
存储桶操作	361
对象操作	368
预签名 URL	378
跨区域访问	386
校验和	387
使用高性能 S3 客户端	389

传输文件和目录	391
S3 事件通知	398
Amazon SNS	405
创建主题	405
列出 Amazon SNS 主题	406
为终端节点订阅主题	407
向主题发布消息	408
为终端节点取消订阅主题	409
删除主题	410
Amazon SQS	411
队列操作	411
消息操作	414
Amazon Transcribe	418
设置麦克风	418
创建发布者	418
创建客户端和启动流	421
更多信息	417
代码示例	424
操作和场景	424
API 网关	426
Application Auto Scaling	430
应用程序恢复控制器	439
Aurora	441
Auto Scaling	476
Amazon Bedrock	538
Amazon Bedrock 运行时系统	544
CloudFront	622
CloudWatch	642
CloudWatch 活动	692
CloudWatch 日志	698
Amazon Cognito Identity	708
Amazon Cognito 身份提供者	715
Amazon Comprehend	742
DynamoDB	753
亚马逊 EC2	832
亚马逊 ECR	906

亚马逊 ECS	945
Elastic Load Balancing-版本 2	959
MediaStore	1003
OpenSearch 服务	1018
EventBridge	1026
预测	1060
AWS Glue	1074
HealthImaging	1097
IAM	1128
AWS IoT	1211
AWS IoT data	1251
Amazon Keyspaces	1254
Kinesis	1280
AWS KMS	1293
Lambda	1329
AWS Marketplace 协议服务	1357
AWS Marketplace Catalog API	1407
MediaConvert	1582
Migration Hub	1604
Amazon Personalize	1617
Amazon Personalize Events	1647
Amazon Personalize Runtime	1650
Amazon Pinpoint	1655
亚马逊 Pinpoint SMS 和语音 API	1699
Amazon Polly	1702
亚马逊 RDS	1708
Amazon Redshift	1751
Amazon Rekognition	1777
Route 53 域注册	1844
Amazon S3	1867
Amazon S3 控件	2009
S3 Glacier	2046
SageMaker	2062
Secrets Manager	2091
亚马逊 SES	2093
亚马逊 SES API v2	2105

亚马逊 SNS	2124
亚马逊 SQS	2189
Step Functions	2230
AWS STS	2254
AWS Support	2257
Systems Manager	2280
Amazon Textract	2322
Amazon Transcribe	2333
跨服务示例	2349
构建应用程序以将数据提交到 DynamoDB 表	2350
构建 Amazon Lex 聊天机器人	2350
构建 Amazon SNS 应用程序	2350
创建消息收发应用程序	2351
创建无服务器应用程序来管理照片	2351
创建 Web 应用程序来跟踪 DynamoDB 数据	2352
创建 Web 应用程序来跟踪 Amazon Redshift 数据	2352
创建 Aurora Serverless 工作项跟踪器	2352
创建用于分析客户反馈的应用程序	2353
PPE在图像中检测	2353
检测图像中的对象	2354
检测视频中的人物和对象	2354
监控 DynamoDB 性能	2355
使用API网关调用 Lambda 函数	2355
使用 Step Functions 调用 Lambda 函数	2355
使用计划的事件调用 Lambda 函数	2356
安全性	2357
数据保护	2357
传输层安全 (TLS)	2358
检查TLS版本	2358
强制执行TLS版本	2359
迁移到 TLS 1.2	2359
Identity and Access Management	2359
受众	2360
使用身份进行身份验证	2360
使用策略管理访问	2363
如何 AWS 服务 使用 IAM	2365

对 AWS 身份和访问进行故障排除	2365
合规性验证	2366
弹性	2367
基础架构安全性	2368
迁移到版本 2	2369
版本 2 中有哪些新功能	2369
S tep-by-step 指令	2370
步骤概述	2370
迁移示例	2371
ArtifactID 映射的包名	2382
命名惯例	2382
异常	2382
1.x 和 2.x 之间的差异	2386
软件包名称更改	2386
将版本 2.x 添加到您的项目	2387
不可变 POJOs	2388
设置器和获取器方法	2388
模型类名	2389
库和实用工具	2389
客户端更改	2391
凭证提供程序更改	2435
地区变化	2443
操作、请求和响应变更	2444
Exception 更改	2446
序列化更改	2447
特定于服务的更改	2448
配置文件更改	2453
外部配置	2454
Waiter	2457
S3 Transfer Manager	2461
EC2 元数据实用程序	2467
CloudFront 预先签名	2475
S3 URI 解析	2478
IAM policy 生成器 API	2481
DynamoDB 映射/文档 APIs	2487
S3 事件通知	2511

并行使用适用于 Java 的 SDK 1.x 和 2.x 版	2518
OpenPGP 密钥	2519
当前密钥	2519
文档历史记录	2521
.....	mmdxxviii

开发人员指南 – AWS SDK for Java 2.x

AWS SDK for Java 提供适用于 AWS 服务的 Java API。利用此 SDK，您可以构建使用 Amazon S3、Amazon EC2、DynamoDB 等的 Java 应用程序。

AWS SDK for Java 2.x 是对版本 1.x 代码库的重大改写。它基于 Java 8+ 构建，并增加了几个请求次数较多的功能。其中包括对非阻塞 I/O 的支持以及在运行时插入不同 HTTP 实现的功能。

我们将定期向 AWS SDK for Java 添加对新服务的支持。有关特定版本中的更改和功能的列表，请参阅 [更改日志](#)。

开始使用 SDK

如果您已准备好亲身体会 SDK，请按照 [入门教程](#) 教程进行操作。

要设置开发环境，请参阅 [设置](#)。

如果您当前使用的是 1.x 版的 SDK for Java，请参阅 [迁移到版本 2](#) 以获取具体指导。

有关向 Amazon S3、DynamoDB、Amazon EC2 和其他 AWS 服务提出请求的信息，请参阅 [使用 SDK for Java](#) 和 [使用 AWS 服务](#)。

开发移动应用程序

如果您是移动应用程序开发人员，Amazon Web Services 提供了 [AWS Amplify](#) 框架。

SDK 主要版本的维护和支持

有关维护和支持 SDK 主要版本及其底层依赖项的信息，请参阅 [《AWS SDKs and Tools Reference Guide》](#) 中的以下主题：

- [AWS SDKs and Tools Maintenance Policy](#)
- [AWS SDKs and Tools Version Support Matrix](#)

其他资源

除了本指南外，还有以下适用于 AWS SDK for Java 开发人员的有价值的在线资源：

- [AWS SDK for Java 2.x API Reference](#)
- [Java 开发人员博客](#)
- [AWS re:Post 中的 Java 开发主题](#)
- GitHub 上的 [SDK 源代码](#)
- [AWS SDK 代码示例库](#)
- [@awsforjava \(Twitter\)](#)

为 SDK 做贡献

开发人员还可以通过以下渠道提供反馈：

- 在 GitHub 上提交问题：
 - [提交《开发人员指南》文档存在的问题](#)
 - [提交开发工具包问题](#)
- 在 AWS SDK for Java 2.x [Gitter 频道](#)加入有关 SDK 的非正式聊天

开始使用 AWS SDK for Java 2.x

为 Amazon Web Services (AWS) AWS SDK for Java 2.x 提供了 Java API。使用 SDK，您可以构建与、Amazon S3 Amazon EC2 DynamoDB、等配合使用的 Java 应用程序。

本教程向您展示了如何使用 [Apache Maven](#) 为适用于 Java 的 SDK 2.x 定义依赖项，然后编写用于连接 Amazon S3 以上传文件的代码。

要完成本教程，请执行以下步骤：

- [步骤 1：为本教程进行设置](#)
- [步骤 2：创建项目](#)
- [步骤 3：编写代码](#)
- [步骤 4：构建并运行应用程序](#)

步骤 1：为本教程进行设置

在开始本教程之前，您需要具备：

- 访问权限 Amazon S3
- 配置为 AWS 服务 使用单点登录访问的 Java 开发环境 AWS IAM Identity Center

请按照 [???](#) 中的说明为本教程进行设置。在[将开发环境配置为 Java SDK 的单点登录访问权限并且 AWS 访问门户会话处于活动状态](#)后，请继续本教程的步骤 2。

步骤 2：创建项目

要为本教程创建项目，您需要运行一条 Maven 命令，该命令会提示您输入有关如何配置项目的信息。完成所有输入并进行确认后，Maven 通过创建 pom.xml 完成项目构建，并创建存根 Java 文件。

1. 打开终端或命令提示符窗口，然后导航到您选择的目录，例如您的 Desktop 或 Home 文件夹。
2. 在终端中输入以下命令，然后按 Enter。

```
mvn archetype:generate \  
-DarchetypeGroupId=software.amazon.awssdk \  
-DarchetypeArtifactId=archetype-app-quickstart \  
-DarchetypeVersion=2.17.100
```

```
-DarchetypeVersion=2.20.43
```

3. 为每个提示输入第二列中列出的值。

提示	要输入的值
Define value for property 'service':	s3
Define value for property 'httpClient':	apache-client
Define value for property 'nativeImage':	false
Define value for property 'credentialProvider':	identity-center
Define value for property 'groupId':	org.example
Define value for property 'artifactId':	getstarted
Define value for property 'version' 1.0-SNAPSHOT:	<Enter>
Define value for property 'package' org.example:	<Enter>

4. 输入最后一个值后，Maven 会列出您所做的选择。通过输入 *Y* 进行确认，或者通过输入 *N* 重新输入值。

Maven 会根据您输入的 `artifactId` 值创建名为 `getstarted` 的项目文件夹。在 `getstarted` 文件夹中，找到一个可以查看的 `README.md` 文件、一个 `pom.xml` 文件和一个 `src` 目录。

Maven 构建了以下目录树。

```
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
```

下面显示的是 pom.xml 项目文件的内容。

pom.xml

dependencyManagement 部分包含一个 AWS SDK for Java 2.x 依赖项，而 dependencies 部分包含一个 Amazon S3 依赖项。由于 maven.compiler.source 和 maven.compiler.target 属性中的值是 1.8，所以该项目使用 Java 1.8。

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

  <groupId>org.example</groupId>
  <artifactId>getstarted</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
    <maven.shade.plugin.version>3.2.1</maven.shade.plugin.version>
    <maven.compiler.plugin.version>3.6.1</maven.compiler.plugin.version>
```

```

    <exec-maven-plugin.version>1.6.0</exec-maven-plugin.version>
    <aws.java.sdk.version>2.20.43</aws.java.sdk.version> <----- SDK version
picked up from archetype version.
    <slf4j.version>1.7.28</slf4j.version>
    <junit5.version>5.8.1</junit5.version>
</properties>

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

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

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

  <dependency>
    <groupId>software.amazon.awssdk</groupId>

```

```
    <artifactId>ssoidc</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>
```

步骤 3：编写代码

以下代码显示的是 Maven 创建的 App 类。main 方法是应用程序的入口点，它会创建 Handler 类的实例，然后调用其 sendRequest 方法。

App 类

```
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");
    }
}
```

Maven 创建的 DependencyFactory 类包含用于构建和返回 [S3Client](#) 实例的 s3Client 工厂方法。S3Client 实例使用基于 Apache 的 HTTP 客户端的实例。这是因为您在 Maven 提示您输入使用哪个 HTTP 客户端时指定了 apache-client。

以下代码中显示了 `DependencyFactory`。

DependencyFactory 类

```
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` 类包含程序的主要逻辑。在 `App` 类中创建 `Handler` 的实例时，`DependencyFactory` 将提供 `S3Client` 服务客户端。您的代码使用 `S3Client` 实例调用 Amazon S3 服务。

Maven 生成以下带有 `TODO` 注释的 `Handler` 类。本教程的下一步会将 `TODO` 替换为代码。

Maven 生成的 Handler 类

```
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.
    }
}
```

要填写逻辑，请将该 `Handler` 类的全部内容替换为以下代码。这将填写 `sendRequest` 方法并添加必要的导入。

实现的 `Handler` 类

该代码首先创建一个新的 S3 桶，其名称的最后一部分使用 `System.currentTimeMillis()` 生成，以使桶名称具有唯一性。

使用 `createBucket()` 方法创建桶后，程序将使用 `S3Client` 的 [putObject](#) 方法上传对象。对象的内容是使用 `RequestBody.fromString` 方法创建的简单字符串。

最后，程序使用 `cleanUp` 方法删除对象，然后删除桶。

```
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.waitFor().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");
}
}
```

步骤 4：构建并运行应用程序

在创建了包含完整 Handler 类的项目后，生成并运行该应用程序。

1. 确保您的 IAM Identity Center 会话处于活动状态。为此，请运行 AWS Command Line Interface 命令 `aws sts get-caller-identity` 并检查响应。如果您没有活动会话，请参阅[此部分](#)了解说明。
2. 打开终端或命令提示符窗口并导航至您的项目目录 `getstarted`。
3. 使用以下命令生成项目：

```
mvn clean package
```

4. 使用以下命令运行应用程序。

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

要查看程序创建的新桶和对象，请执行以下步骤。

1. 在 `Handler.java` 中，注释掉 `sendRequest` 方法中的 `cleanUp(s3Client, bucket, key)` 行并保存文件。
2. 运行 `mvn clean package` 以重新生成项目。
3. 重新运行 `mvn exec:java -Dexec.mainClass="org.example.App"` 以再次上传文本对象。

4. 登录 [S3 控制台](#)，在新创建的桶中查看新对象。

查看文件后，删除对象，然后删除桶。

成功

如果您的 Maven 项目生成和运行都没有错误，那么恭喜您！您已经使用适用于 Java 的 SDK 2.x 成功构建了您的第一个 Java 应用程序。

清理

要清除您在本教程中创建的资源，请执行以下操作：

- 在 [S3 控制台](#) 中删除运行应用程序时创建的所有对象和所有桶（如果您尚未执行此操作）。
- 删除项目文件夹 (getstarted)。

后续步骤

现在您已掌握了基础知识，接下来，您可以了解以下内容：

- [与... 合作 Amazon S3](#)
- [使用其他](#)（例如 Amazon Web Services [DynamoDB](#) [Amazon EC2](#)、）和 [各种数据库服务](#)
- [使用 SDK](#)
- [安全性 AWS SDK for Java](#)

设置 AWS SDK for Java 2.x

本部分提供有关如何设置开发环境和项目以使用 AWS SDK for Java 2.x 的信息。

设置概述

要成功开发 AWS 服务 使用访问的应用程序 AWS SDK for Java ，需要满足以下条件：

- Java SDK 必须有权访问凭据才能代表您[对请求进行身份验证](#)。
- 为软件开发工具包配置的 [IAM 角色的权限](#) 必须允许访问您的应用程序所需的权限。AWS 服务与 PowerUserAccess AWS 托管策略关联的权限足以满足大多数开发需求。
- 包含以下元素的开发环境：
 - 通过以下方式中的至少一种方式设置的 [共享配置文件](#)：
 - 该 config 文件包含 [IAM Identity Center 单点登录设置](#)，以便软件开发工具包可以获取 AWS 证书。
 - credentials 文件包含临时凭证。
 - [安装了 Java 8 或更高版本](#)。
 - 一种 [构建自动化工具](#)，例如 [Maven](#) 或 [Gradle](#)。
 - 用于处理代码的文本编辑器。
 - (可选，但建议使用) IDE (集成开发环境)，例如 [IntelliJ ID EA](#)、[Eclipse](#) 或 [NetBeans](#)

使用 IDE 时，还可以集成 AWS Toolkit，以便更轻松地使用 AWS 服务。[AWS Toolkit for IntelliJ](#) 和 [AWS Toolkit for Eclipse](#) 是两个可用于 Java 开发的工具包。

- 准备好运行应用程序时处于活动状态的 AWS 访问门户会话。您可以使用 [启动 IAM Identity Center AWS 访问门户的登录流程](#)。AWS Command Line Interface

Important

本设置部分中的说明假设您或组织使用 IAM Identity Center。如果您的组织使用独立于 IAM Identity Center 运行的外部身份提供商，请了解如何获取临时凭证以供适用于 Java 的 SDK 使用。按照 [以下说明](#) 向 ~/.aws/credentials 文件添加临时凭证。

如果您的身份提供商自动向 `~/.aws/credentials` 文件添加临时凭证，请确保配置文件名称为 `[default]`，这样您就无需向 SDK 或 AWS CLI 提供配置文件名称。

设置身份验证

AWS SDK [和工具参考指南中的身份验证和访问](#) 主题描述了不同的身份验证选项。我们建议您按照说明 [设置对 IAM Identity Center 的访问权限](#)，以便软件开发工具包可以获取证书。按照说明操作后，[您的系统已设置为](#) 允许 SDK 对请求进行身份验证。

设置 SDK 的单点登录访问权限

在完成 [编程访问部分](#) 的步骤 2 以便 SDK 可以使用 IAM Identity Center 身份验证后，您的系统应包含以下元素。

- AWS CLI，用于在运行应用程序之前启动 [AWS 访问门户会话](#)。
- 包含 [默认配置文件](#) 的 `~/.aws/config` 文件。在向 AWS 发送请求之前，适用于 Java 的 SDK 使用配置文件的 SSO 令牌提供程序配置来获取凭证。该 `sso_role_name` 值是与 IAM Identity Center 权限集关联的 IAM 角色，应允许访问您的应用程序中 AWS 服务使用的权限。

以下示例 `config` 文件展示了使用 SSO 令牌提供程序配置来设置的默认配置文件。配置文件的 `sso_session` 设置是指所指定的 `sso-session` 节。该 `sso-session` 部分包含启动 AWS 访问门户会话的设置。

```
[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
```

有关 SSO 令牌提供程序配置中使用的设置的更多详细信息，请参阅《AWS SDKs and Tools Reference Guide》中的 [SSO token provider configuration](#)。

如果您的开发环境未如前所示进行编程访问设置，请按照 [《SDK 参考指南》中的步骤 2](#) 进行操作。

使用登录 AWS CLI

在运行可访问的应用程序之前 AWS 服务，您需要进行有效的 AWS 访问门户会话，这样 SDK 才能使用 IAM Identity Center 身份验证来解析证书。在中运行以下命令登录 AWS CLI AWS 访问门户。

```
aws sso login
```

由于您有默认的配置文件设置，因此无需使用 `--profile` 选项调用该命令。如果您的 SSO 令牌提供程序配置在使用指定的配置文件，则命令为 `aws sso login --profile named-profile`。

要测试是否已有活动会话，请运行以下 AWS CLI 命令。

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

对此命令的响应应该报告共享 config 文件中配置的 IAM Identity Center 账户和权限集。

Note

如果您已经有一个有效的 AWS 访问门户会话并且 `aws sso login` 正在运行，则无需提供凭据。

但是，您将看到一个对话框，请求 `botocore` 访问您的信息的权限。`botocore` 是 AWS CLI 的基础。

选择“允许”以授权访问您的信息，AWS CLI 以及适用于 Java 的 SDK。

安装 Java 和构建工具

您的开发环境需要以下组件：

- Java 8 或更高版本。AWS SDK for Java [它可与 Oracle Java SE 开发套件以及红帽 OpenJDK 和 Adoptium Amazon Corretto 等开放 Java 开发套件 \(OpenJDK\) 的发行版配合使用。](#)
- 支持 Maven Central 的构建工具或 IDE，例如 Apache Maven、Gradle 或 IntelliJ。
 - 有关如何安装和使用 Maven 的信息，请参阅 <https://maven.apache.org/>。
 - 有关如何安装和使用 Gradle 的信息，请访问 <https://gradle.org/>。
 - 有关如何安装和使用 IntelliJ IDEA 的信息，请访问 <https://www.jetbrains.com/idea/>。

其他身份验证选项

有关 SDK 身份验证的更多选项，例如配置文件和环境变量的使用，请参阅 AWS SDK 和工具参考指南中的[配置](#)章节。

设置 Apache Maven 项目

您可以使用 [Apache Maven](#) 设置和构建 AWS SDK for Java 项目或[构建 SDK 本身](#)。

先决条件

要将 AWS SDK for Java 与 Maven 结合使用，您需要以下各项：

- Java 8.0 或更高版本。您可以从 <http://www.oracle.com/technetwork/java/javase/downloads/> 下载最新的 Java SE 开发工具包软件。AWS SDK for Java 还可以与 [OpenJDK](#) 和 Amazon Corretto (开源 Java 开发工具包 (OpenJDK) 的一个发行版) 结合使用。从 <https://openjdk.java.net/install/index.html> 下载最新 OpenJDK 版本。从 [Corretto 页面](#) 下载最新的 Amazon Corretto 8 或 Amazon Corretto 11 版本。
- Apache Maven。如果您需要安装 Maven，请转到 <http://maven.apache.org/> 下载并安装它。

创建 Maven 项目

要从命令行创建 Maven 项目，请从终端或命令提示符窗口运行以下命令。

```
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

将 `com.example.myapp` 替换为您的应用程序的完整程序包命名空间。还可以将 `myapp` 替换为您的项目名称。这将成为项目的目录的名称。

要使用最新版本的原型，请将 `2.X.X` 替换为 [Maven Central 中的最新版本](#)。

此命令使用原型模板工具包创建 Maven 项目。原型为 AWS Lambda 函数处理程序项目生成支架。此项目原型预配置为使用 Java SE 8 进行编译，并包括适用于 Java 的 SDK 2.x 版本的依赖项（用 `-DarchetypeVersion` 指定）。

有关创建和配置 Maven 项目的更多信息，请参阅 [Maven 入门指南](#)。

为 Maven 配置 Java 编译器

如果您使用前面所述的 AWS Lambda 项目原型创建了项目，则已经为您完成了 Java 编译器的配置。

要验证此配置存在，请首先从执行上一个命令时创建的项目文件夹（例如，`myapp`）中打开 `pom.xml` 文件。检查第 11 行和第 12 行以查看此 Maven 项目的 Java 编译器版本设置，以及检查第 71-75 行上是否包含所需的 Maven 编译器插件。

```
<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>
```

如果使用其他原型或其他方法创建项目，则必须确保 Maven 编译器插件是构建的一部分，并且将 `pom.xml` 文件中的源和目标属性均设置为 1.8。

有关一种配置这些必需设置的方法，请参阅上一个代码段。

或者，您可以使用插件声明内联配置编译器配置，如下所示。

```
<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>
```

将开发工具包声明为依赖项

要在项目中使用 AWS SDK for Java，您需要在项目的 `pom.xml` 文件中将该工具包声明为依赖项。

如果您使用前面所述的项目原型创建了项目，则已将 SDK 的最新版本配置为项目中的依赖项。

原型为 `software.amazon.awssdk` 组 ID 生成一个 BOM（材料清单）构件依赖项。使用 BOM 时，您不必为共享相同组 ID 的各个构件依赖项指定 maven 版本。

如果您以不同的方式创建了 Maven 项目，请通过确保 `pom.xml` 文件包含以下内容来为项目配置最新版本的开发工具包。

```
<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

将 `pom.xml` 文件中的 `2.X.X` 替换为 [AWS SDK for Java 2.x 的最新版本](#)。

设置开发工具包模块的依赖项

现在您已配置了开发工具包，您可以为项目中要使用的一个或多个AWS SDK for Java模块添加依赖项。

尽管您可以为每个组件指定版本号，但无需这样做，因为您已使用材料清单构件在 `dependencyManagement` 部分中声明了 SDK 版本。要加载给定模块的其他版本，请为其依赖项指定版本号。

如果您使用前面所述的项目原型创建了项目，则您的项目已配置了多个依赖项。其中包括 AWS Lambda 函数处理程序和 Amazon S3 的依赖项，如下所示。

```
<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

在上面的 `pom.xml` 示例中，依赖项来自不同的 `groupId`。s3 依赖项来自 `software.amazon.awssdk`，而 `aws-lambda-java-core` 依赖项来自 `com.amazonaws`。BOM 依赖项管理配置会影响 `software.amazon.awssdk` 的构件，因此需要为 `aws-lambda-java-core` 构件提供一个版本。

对于使用适用于 Java 的 SDK 2.x 开发的 Lambda 函数处理程序，正确的依赖项是 `aws-lambda-java-core`。但是，如果您的应用程序需要使用诸如 `listFunctions`、`deleteFunction`、`invokeFunction` 和 `createFunction` 之类的操作来管理 Lambda 资源，则您的应用程序需要以下依赖项。

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

Note

s3 依赖项与 `netty-nio-client` 和 `apache-client` 传递依赖项相斥。原型中包含了 `url-connection-client` 依赖项来代替这两个 HTTP 客户端，这有助于[减少 AWS Lambda 函数的启动延迟](#)。

将模块添加到您的项目中，以获得您的项目所需的 AWS 服务和功能。由 AWS SDK for Java BOM 管理的模块（依赖项）列在 [Maven Central 存储库](#) 中。

Note

您可以从代码示例中查看 `pom.xml` 文件，以确定您的项目需要哪些依赖项。例如，如果您对 DynamoDB 服务的依赖项感兴趣，请在 GitHub 上的 [AWS 代码示例存储库](#) 中查看[此示例](#)。（在 [/javav2/example_code/dynamodb](#) 下查找 `pom.xml` 文件。）

将整个开发工具包构建到您的项目中

为了优化您的应用程序，强烈建议您仅拉入所需的组件而不是整个开发工具包。但是，要将整个 AWS SDK for Java 构建到您的项目中，请在 `pom.xml` 文件中声明它，如下所示。

```
<project>
```

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

重新构建项目。

配置 pom.xml 文件后，您可以使用 Maven 构建您的项目。

要从命令行生成 Maven 项目，请打开终端或命令提示符窗口，导航到项目目录（例如，myapp），输入或粘贴以下命令，然后按 Enter 或 Return。

```
mvn package
```

这将在 target 目录（例如，myapp/target）中创建单个 .jar 文件 (JAR)。此 JAR 包含您在 pom.xml 文件中指定为依赖项的所有开发工具包模块。

设置 Gradle 项目

您可以使用 [Gradle](#) 来设置和构建 AWS SDK for Java 项目。

以下示例中的初始步骤来自 [Gradle 8.4 版的入门指南](#)。如果您使用的是其他版本，结果可能会略有不同。

使用 Gradle 创建 Java 应用程序（命令行）

1. 创建一个用于存放项目的目录。在此示例中，该目录名为 demo。
2. 在 demo 目录中，执行 gradle init 命令并提供以红色突出显示的值，如以下命令行输出所示。在演练中，我们选择 Kotlin 作为构建脚本 DSL 语言，但本主题末尾还提供了使用 Groovy 的完整示例。

```
> 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` 任务完成后，`demo` 目录包含以下树结构。在下一个部分中，我们将仔细研究主构建文件 `build.gradle.kts`（以红色突出显示）。

```
### 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 文件包含以下支架式内容。

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

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

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



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

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

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

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

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

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

4. 使用支架式 Gradle 构建文件作为 AWS 项目的基础。
 - a. 要管理 Gradle 项目的 SDK 依赖项，请将 AWS SDK for Java 2.x 的 Maven 材料清单 (BOM) 添加到 `build.gradle.kts` 文件中的 `dependencies` 部分。

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

Note

在此示例构建文件中，将 2.21.1 替换为适用于 Java 的 SDK 2.x 的最新版本。在 [Maven Central 存储库](#) 中查找可用的最新版本。

- b. 在 `dependencies` 部分中指定您的应用程序需要的 SDK 模块。例如，以下内容添加了对 Amazon Simple Storage Service 的依赖项。

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

Gradle 会自动使用 BOM 中的信息来解析所声明依赖项的正确版本。

以下示例显示了使用 Kotlin 和 Groovy DSL 的完整 Gradle 构建文件。构建文件包含 Amazon S3、身份验证、日志和测试的依赖项。Java 的源版本和目标版本均为版本 11。

Kotlin DSL (build.gradle.kts)

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

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

repositories {
```

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

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

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

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

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

Groovy DSL (build.gradle)

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

```
*/

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

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

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

有关后续步骤，请参阅 Gradle 网站上的入门指南，了解有关如何[构建和运行 Gradle 应用程序](#)的说明。

为 AWS SDK for Java 设置 GraalVM 原生映像项目

在 2.16.1 及更高版本中，AWS SDK for Java 为 GraalVM 原生映像应用程序提供了开箱即用的支持。使用 archetype-app-quickstart Maven 原型设置具有内置原生映像支持的项目。

先决条件

- 完成[设置 AWS SDK for Java 2.x](#) 中的步骤。
- 安装 [GraalVM 原生映像](#)。

使用原型创建项目

要创建具有内置原生映像支持的 Maven 项目，请在终端或命令提示符窗口中使用以下命令。

Note

将 `com.example.mynativeimageapp` 替换为您的应用程序的完整程序包命名空间。还要将 `mynativeimageapp` 替换为您的项目名称。这将成为项目的目录的名称。

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

此命令创建一个 Maven 项目，该项目配置了 AWS SDK for Java、Amazon S3 和 ApacheHttpClient HTTP 客户端的依赖项。它还包含 [GraalVM 原生映像 Maven 插件](#) 的依赖项，使您可以使用 Maven 构建原生映像。

要包含其他 Amazon Web Services 的依赖项，请将 `-Dservice` 参数的值设置为该服务的构件 ID。示例包括 `dynamodb`、`comprehend` 和 `pinpoint`。有关构件 ID 的完整列表，请参阅 Maven Central 中的 [software.amazon.awssdk](https://mvnrepository.com/software.amazon.awssdk) 托管依赖项列表。

要使用异步 HTTP 客户端，请将 `-DhttpClient` 参数设置为 `netty-nio-client`。要使用 `URLConnectionHttpClient` 作为同步 HTTP 客户端，而不使用 `apache-client`，请将 `-DhttpClient` 参数设置为 `url-connection-client`。

生成原生映像

创建项目后，从项目目录（例如 `mynativeimageapp`）中运行以下命令：

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

这将在 `target` 目录（例如 `target/mynativeimageapp`）中创建原生映像应用程序。

使用 AWS SDK for Java 2.x

完成[设置软件开发工具包中的步骤](#)后，您就可以向诸如 [Amazon S3](#)、[DynamoDB](#)、IAM、Amazon EC2 等 AWS 服务提出请求了。

使用服务客户端

创建服务客户端

要向发出请求 AWS 服务，必须先使用静态工厂方法为该服务实例化服务客户端。builder() 该 builder() 方法返回一个允许您自定义服务客户端的 builder 对象。常用的 setter 方法会返回 builder 对象，由此可以将方法调用组合起来，这样不仅方便而且代码更加便于阅读。在配置了所需属性后，可以调用 build() 方法创建客户端。

例如，以下代码段将 Ec2Client 对象实例化为 Amazon EC2 的服务客户端。

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

Note

开发工具包中的服务客户端是线程安全的。为了获得最佳性能，应将其作为永久对象。每个客户端自己有连接池资源，当客户端收集到垃圾时相应资源会释放。

服务客户端对象是不可变的，因此您必须为向其发出请求的每个服务创建一个新的客户端，或者，如果您希望使用不同的配置向同一服务发出请求，也需要创建一个新的客户端。

并非所有 AWS 服务都需要 Region 在服务客户端生成器中指定；但是，最佳做法是为在应用程序中进行的 API 调用设置区域。有关更多信息，请参阅 [AWS 区域选择](#)。

默认客户端配置

客户端生成器包含名为 create() 的另一个工厂方法。此方法将使用默认配置创建服务客户端。该客户端使用默认提供程序链加载凭证和 AWS 区域区域。如果不能根据运行应用程序的环境确定凭证或区域，则对 create 的调用失败。有关 SDK 如何确定要使用的凭证和区域的更多信息，请参阅 [使用凭证](#) 和 [区域选择](#)。

例如，以下代码段将 `DynamoDbClient` 对象实例化为 Amazon DynamoDB 的服务客户端。

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

配置服务客户端

要自定义服务客户端的配置，请使用 `builder()` 工厂方法上的 `setter`。为了方便起见并创建更具可读性的代码，请将方法链接起来以设置多个配置选项。

以下示例显示了配置了多个自定义设置的 `S3Client`。

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

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

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

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

提出请求

使用服务客户端向对应的发出请求 AWS 服务。

例如，以下代码段演示如何创建 `RunInstancesRequest` 对象以创建新的 Amazon EC2 实例：

```
// 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);
```

SDK 并不创建请求并在实例中传递，而是提供可用于创建请求的生成器。借助生成器，您可以使用 Java lambda 表达式创建请求“内联”。

以下示例使用通过[生成器创建请求的 runInstances 方法](#)版本重写了前面的示例。

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

处理响应

对于大多数服务操作，SDK 会返回响应对象。您的代码可以根据需要处理响应对象中的信息。

例如，以下代码片段打印出上一个请求中随[RunInstancesResponse](#)对象返回的第一个实例 ID。

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

但是，并非所有操作都会返回包含服务特定数据的响应对象。在这些情况下，您可以查询 HTTP 响应状态以了解操作是否成功。

例如，以下代码段中的代码会检查 HTTP 响应，以查看 Amazon 简单电子邮件服务的[DeleteContactList](#)操作是否成功。

```
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());
}
```

关闭服务客户端

作为最佳实践，您应该在应用程序的生命周期内使用服务客户端进行多个 API 服务调用。但是，如果您需要一次性使用服务客户端，或者不再需要该服务客户端，请将其关闭。

当不再需要服务客户端时，请调用 `close()` 方法，以释放资源。

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

如果您需要一次性使用服务客户端，则可以通过 `try-with-resources` 语句将服务客户端实例化为资源。服务客户端将实现 [Autoclosable](#) 接口，因此 JDK 会在语句末尾自动调用 `close()` 方法。

以下示例演示如何使用服务客户端进行一次性调用。调用 `StsClient` AWS Security Token Service 的，将在返回账户 ID 后关闭。

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

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

处理异常

SDK 使用运行时系统异常（或未选中的异常），为您提供对错误处理的精细控制，并确保异常处理会随着您的应用程序而扩展。

一个 [SdkServiceException](#)，或其子类之一，是 SDK 会引发的最常见异常形式。这些异常表示来自 AWS 服务的响应。您还可以处理在客户端（即开发或应用程序环境中）出现问题（例如网络连接故障）时发生的 [SdkClientException](#)。

此代码段演示了将文件上传到 Amazon S3 时处理服务异常的一种方法。该示例代码可捕获客户端和服务端异常，记录详细信息并退出应用程序。

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

try {

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

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

} catch (S3Exception se) {
    System.err.println("Service exception thrown.");
    System.err.println(se.awsErrorDetails().errorMessage());
} catch (SdkClientException ce){
    System.err.println("Client exception thrown.");
    System.err.println(ce.getMessage());
} finally {
    System.exit(1);
}
```

有关更多信息，请参阅[处理异常](#)。

使用 waiter

有些请求需要时间才能处理，例如在中创建新表 DynamoDB 或创建新 Amazon S3 存储桶。要确保资源在代码继续运行之前准备就绪，请使用 Waiter。

例如，以下代码片段在中创建了一个新表（“MyTable”）DynamoDB，等待该表ACTIVE处于状态，然后打印出响应：

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

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

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

有关更多信息，请参阅[使用 waiter](#)。

HTTP 客户端

在使用 AWS SDK for Java 构建的应用程序中，您可以更改 HTTP 客户端的默认配置。有关如何配置 HTTP 客户端和设置的信息，请参阅[HTTP 配置](#)。

超时

您可以使用[apiCallTimeout](#)和的[apiCallAttemptTimeout](#)设置器为每个服务客户端配置超时。[ClientOverrideConfiguration.Builder](#)的[apiCallTimeout](#)设置是允许客户端完成 API 调用的执行所需的时间。该[apiCallAttemptTimeout](#)设置是在放弃之前等待每个 HTTP 请求（重试）完成的时间量。

以下示例为 S3 客户端设置了两个超时时间。

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

您还可以在请求级别设置超时，方法是配置[AwsRequestOverrideConfiguration](#)并使用[overrideConfiguration](#)方法将其提供给请求对象。

以下示例对 S3 PutObject 操作使用相同的超时设置，但处于请求级别。

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

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

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

```
RequestBody.fromString("test"));
```

执行拦截器

您可以编写代码，在请求/响应生命周期的不同阶段拦截 API 请求和响应的执行。这使您能够发布指标、修改正在进行的请求、调试请求处理、查看异常等。有关更多信息，请参阅 [《AWS SDK for Java API 参考》中的 `ExecutionInterceptor` 接口](#)。

其他信息

- 有关上述代码片段的完整示例，请参阅 [使用 Amazon DynamoDB Amazon EC2](#)、[使用和使用 Amazon S3](#)。

向 SDK 提供临时凭证

在请求 Amazon Web Services 使用之前 AWS SDK for Java 2.x，SDK 会对颁发的临时证书进行 AWS 加密签名。要访问临时凭证，SDK 会通过检查多个位置来检索配置值。

本主题讨论了使 SDK 能够访问临时凭证的几种方式。

主题

- [配置对临时凭证的访问权限](#)
- [默认凭证提供程序链](#)
- [使用特定的凭证提供程序或提供程序链](#)
- [使用配置文件](#)
- [从外部流程加载临时凭证](#)
- [在代码中提供临时凭证](#)
- [阅读上的 IAM 角色证书 Amazon EC2](#)

配置对临时凭证的访问权限

为了提高安全性，AWS 建议您将适用于 Java 的 SDK 配置为 [使用临时凭证](#) 而不是长期证书。临时凭证由访问密钥（访问密钥 ID 与秘密访问密钥）和会话令牌组成。我们建议您 [配置 SDK](#) 以自动获取临时凭证，因为令牌刷新过程是自动的。但您也可以直接向 [SDK 提供临时凭证](#)。

IAM Identity Center 配置

当您将 SDK 配置为使用本指南的[???](#)中所述的 IAM Identity Center 单点登录访问时，SDK 会自动使用临时凭证。

SDK 使用 IAM Identity Center 访问令牌来访问用您的 config 文件中的 sso_role_name 设置配置的 IAM 角色。SDK 代入此 IAM 角色并检索用于 AWS 服务 请求的临时凭证。

有关软件开发工具包如何从配置中获取临时证书的更多详细信息，请参阅软件开发工具包和 AWS 工具参考指南的[了解 IAM Identity Center 身份验证](#)部分。

从 AWS 访问门户检索

作为 IAM Identity Center 单点登录配置的替代方案，您可以复制和使用 AWS 访问门户中提供的临时证书。您可以在配置文件中使用临时凭证，也可以将其用作系统属性和环境变量的值。

为临时凭证设置本地凭证文件

1. [创建共享 credentials 文件](#)
2. 在 credentials 文件中，粘贴以下占位符文本，直到粘贴有效的临时凭证为止。

```
[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. 保存该文件。该 ~/.aws/credentials 文件现在应该存在于您的本地开发系统上。此文件包含[\[默认\] 配置文件](#)，如果未指定特定的命名配置文件，则适用于 Java 的 SDK 将使用该配置文件。
4. [登录 AWS 访问门户](#)
5. 按照[手动刷新凭证](#)标题下的说明从 AWS 访问门户复制 IAM 角色证书。
 - a. 对于链接的说明中的步骤 4，选择可授予访问权限以满足您的开发需求的 IAM 角色名称。此角色的名称通常类似于 PowerUserAccess 或开发人员。
 - b. 对于步骤 7，选择将配置文件手动添加到您的 AWS 凭证文件选项并复制内容。
6. 将复制的凭证粘贴到您的本地 credentials 文件中，并移除生成的配置文件名称。您的文件应类似于以下内容。

```
[default]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
```


- SDK 使用 [EnvironmentVariableCredentialsProvider](#) 类从 `AWS_ACCESS_KEY_ID`、`AWS_SECRET_ACCESS_KEY`、和 `AWS_SESSION_TOKEN` 环境变量加载临时证书。
3. 来自 Web 身份令牌 AWS Security Token Service
- SDK 使用 [WebIdentityTokenFileCredentialsProvider](#) 类从 Java 系统属性或环境变量加载临时证书。
4. 共享的 `credentials` 和 `config` 文件
- 软件开发工具包使用从共享 `credentials` 和文件中的 `[default]` 配置 `config` 文件中加载 IAM Identity Center 单点登录设置或临时证书。 [ProfileCredentialsProvider](#)

AWS 软件开发工具包和工具参考指南 [详细介绍了](#) 适用于 Java 的 SDK 如何与 IAM Identity Center 单点登录令牌配合使用，以获取开发工具包用来调用的临时证书。 AWS 服务

Note

`credentials` 和 `config` 文件由各种 AWS SDK 和工具共享。有关更多信息，请参阅《AWS SDKs and Tools Reference Guide》中的 [.aws/credentials](#) 和 [.aws/config](#) 文件。

5. Amazon ECS 容器凭证
- SDK 使用 [ContainerCredentialsProvider](#) 类从 `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` 系统环境变量加载临时证书。
6. Amazon EC2 实例 IAM 角色提供的证书
- SDK 使用 [InstanceProfileCredentialsProvider](#) 类从 Amazon EC2 元数据服务加载临时证书。

使用特定的凭证提供程序或提供程序链

作为默认凭证提供程序链的替代方案，您可以指定 SDK 应使用哪个凭证提供程序。当您提供特定的凭证提供程序时，SDK 会跳过检查各个位置的过程，这会稍微缩短创建服务客户端的时间。

例如，如果您使用环境变量设置默认配置，请在服务客户端生成器上为该 `credentialsProvider` 方法提供一个 [EnvironmentVariableCredentialsProvider](#) 对象，如以下代码片段所示。

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



```
.build();
```

有关证书提供者和提供者链的完整列表，请参阅中的所有已知实现类。[AwsCredentialsProvider](#)

Note

您可以通过实现 `AwsCredentialsProvider` 接口来使用自己的凭证提供程序或提供程序链。

使用配置文件

使用共享的 `config` 和 `credentials` 文件，您可以设置多个配置文件。这使您的应用程序能够使用多组凭证配置。前面提到的 `[default]` 配置文件。SDK 使用该 [ProfileCredentialsProvider](#) 类从共享 `credentials` 文件中定义的配置文件加载设置。

以下代码片演示如何使用名为 `my_profile` 的配置文件中定义的设置来生成服务客户端。

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

将另一个配置文件设置为默认配置文件

要将 `[default]` 配置文件以外的另一个配置文件设置为应用程序的默认配置文件，请将 `AWS_PROFILE` 环境变量设置为自定义配置文件的名称。

要在 Linux、macOS 或 Unix 上设置这些变量，请使用 `export`：

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

要在 Windows 上设置这些变量，请使用 `set`：

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

或者，将 `aws.profile` Java 系统属性设置为配置文件的名称。

重新加载配置文件凭证

对于其生成器上具有 `profileFile()` 方法的任何凭证提供程序，您都可以将其配置为能够重新加载配置文件凭证。这些凭证配置文件类是：`ProfileCredentialsProvider`、`DefaultCredentialsProvider`、`InstanceProfileCredentialsProvider` 和 `ProfileTokenProvider`。

Note

配置文件凭证的重新加载仅适用于配置文件中的以下设置：`aws_access_key_id`、`aws_secret_access_key` 和 `aws_session_token`。诸如 `region`、`sso_session`、`sso_account_id` 和 `source_profile` 之类的设置将被忽略。

要将支持的凭证提供程序配置为重新加载配置文件设置，请为 `profileFile()` 生成器方法提供一个 [ProfileFileSupplier](#) 实例。以下代码示例演示的 `ProfileCredentialsProvider` 将从 [default] 配置文件重新加载凭证设置。

```
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().
*/
```

调用 `ProfileCredentialsProvider.resolveCredentials()` 时，适用于 Java 的 SDK 会重新加载设置。`ProfileFileSupplier.defaultSupplier()` 是 SDK 提供的 `ProfileFileSupplier` 的[几种便捷实现](#)之一。如果您的用例需要，您可以提供自己的实现。

以下示例演示 `ProfileFileSupplier.reloadWhenModified()` 便利方法的使用。`reloadWhenModified()` 采用一个 `Path` 参数，这使您可以灵活地指定配置的源文件而不是标准 `~/.aws/credentials` (或 `config`) 位置。

仅当 SDK 确定文件内容已修改时，才会在调用 `resolveCredentials()` 时重新加载设置。

```
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()` 方法合并多个配置文件的内容。您可以决定是在每次调用 `resolveCredentials()` 时重新加载文件，还是在首次读取文件时固定其设置。

以下示例演示的 `DefaultCredentialsProvider` 将合并两个包含配置文件设置的文件的设置。每次调用 `resolveCredentials()` 并且设置发生更改时，SDK 都会重新加载 `credentialsFilePath` 变量所指向的文件中的设置。`profileFile` 对象的设置保持不变。

```
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.
*/
```

*/

从外部流程加载临时凭证

Warning

以下内容描述从外部流程获取临时凭证的方法。这可能很危险，因此请谨慎行事。如果可能，应优先选择其他凭证提供者。如果使用此选项，则应确保使用操作系统的安全最佳实践尽可能锁定 config 文件。

确保您的自定义凭证工具不会向 StdErr 中写入任何机密信息。SDK 和 AWS CLI 可以捕获和记录此类信息，从而有可能将其暴露给未经授权的用户。

使用适用于 Java 的 SDK 2.x，您可以从外部流程获取用于自定义用例的临时凭证。可通过两种方式配置此功能。

使用 `credential_process` 设置

如果您有提供临时凭证的方法，则可以通过将 `credential_process` 设置作为配置文件定义的一部分添加到 config 文件中来进行集成。您指定的值必须使用命令文件的完整路径。如果文件路径包含任何空格，则必须用引号将其括起来。

SDK 将完全按照给定的形式调用命令，然后从 stdout 中读取 JSON 数据。

以下示例演示如何对不带空格的文件路径和带空格的文件路径使用此设置。

Linux/macOS

文件路径中没有空格

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

文件路径中有空格

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

Windows

文件路径中没有空格

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

文件路径中有空格

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

以下代码段演示如何生成一个使用名为 `process-credential-profile` 的配置文件中定义的临时凭证的服务客户端。

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

有关使用外部进程作为临时证书来源的详细信息，请参阅《软件开发工具包和 AWS 工具参考指南》中的 [“流程凭证”部分](#)。

使用 `ProcessCredentialsProvider`

除了使用 `config` 文件中的设置之外，您还可以使用 SDK 的 [ProcessCredentialsProvider](#) 通过 Java 加载临时凭证。

以下示例演示如何通过使用 `ProcessCredentialsProvider` 和配置使用临时凭证的服务客户端来指定外部流程的各种版本。

Linux/macOS

文件路径中没有空格

```
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();
```

文件路径中有空格

```
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

文件路径中没有空格

```
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();
```

文件路径中有空格

```
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();
```

在代码中提供临时凭证

如果默认凭证链、特定的或自定义的提供程序或提供程序链都不适用于您的应用程序，您可直接在代码中提供所需的临时凭证。这些证书可以是[上面描述的 IAM 角色证书](#)，也可以是从 AWS Security Token Service (AWS STS) 中检索的临时证书。如果您使用检索临时证书 AWS STS，请将其提供给 AWS 服务客户端，如以下代码示例所示。

1. 通过调用 `StsClient.assumeRole()` 来代入角色。
2. 创建一个[StaticCredentialsProvider](#)对象并为其提供该 `AwsSessionCredentials` 对象。
3. 使用 `StaticCredentialsProvider` 配置服务客户端生成器并生成客户端。

以下示例使用返回的 IAM 代入角色的临时证书创建 Amazon S3 服务客户端。AWS STS

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

权限集

AWS IAM Identity Center 中定义的以下权限集允许身份（用户）执行以下两个操作

1. Amazon Simple Storage Service 的 GetObject 操作。
2. AWS Security Token Service 的 AssumeRole 操作。

如果不代入角色，则示例中显示的 `s3.listBuckets()` 方法将失败。

```
{
  "Version": "2012-10-17",
  "Statement": [
```



```
{
  "Effect": "Allow",
  "Action": [
    "s3:GetObject",
    "sts:AssumeRole"
  ],
  "Resource": [
    "*"
  ]
}
```

担任的角色

代入的角色权限策略

以下权限策略附加到上一个示例中代入的角色。该权限策略允许列出该角色所在账户中的所有桶。

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

代入的角色信任策略

以下信任策略附加到上一示例中代入的角色。该策略允许两个账户中的身份（用户）代入该角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
    "Principal": {
      "AWS": [
        "arn:aws:iam::111122223333:root",
        "arn:aws:iam::555555555555:root"
      ]
    },
    "Action": "sts:AssumeRole",
    "Condition": {}
  }
}
```

阅读上的 IAM 角色证书 Amazon EC2

您可以使用 IAM 角色管理在 EC2 实例上运行并发出 AWS CLI 或 AWS API 请求的应用程序的临时证书。这优先于在 EC2 实例中存储访问密钥。要向 EC2 实例分配 AWS 角色并使其可供其所有应用程序使用，您需要创建附加到该实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅《IAM 用户指南》中的[使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限](#)。

本主题提供有关如何设置您的 Java 应用程序以在 EC2 实例上运行以及如何启用 Java SDK 获取 IAM 角色证书的信息。

从环境中获取 IAM 角色证书

如果您的应用程序使用 `create` 方法（或 `builder().build()` 方法）创建 AWS 服务客户端，则 SDK for Java 将使用默认的凭证提供程序链。默认凭证提供程序链在执行环境中搜索配置元素，SDK 可以用这些元素换取临时证书。[the section called “默认凭证提供程序链”](#) 部分描述了完整的搜索流程。

只有当您的应用程序在 Amazon EC2 实例上运行时，默认提供程序链中的最后一步才可用。在此步骤中，SDK 使用 `InstanceProfileCredentialsProvider` 来读取 EC2 实例配置文件中定义的 IAM 角色。然后，SDK 会获取该 IAM 角色的临时凭证。

尽管这些凭证是临时凭证，而且最终会过期，但 `InstanceProfileCredentialsProvider` 会定期为您刷新它们，保证这些凭证可允许您继续访问 AWS。

以编程方式获取 IAM 角色证书

作为最终使用 EC2 `InstanceProfileCredentialsProvider` 上的默认凭证提供商链的替代方案，您可以使用显式配置服务客户端 `InstanceProfileCredentialsProvider`。以下代码段演示了这种方法。

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

安全获取 IAM 角色证书

默认情况下，EC2 实例运行 [IMDS](#)（实例元数据服务），允许软件开发工具包 `InstanceProfileCredentialsProvider` 访问诸如已配置的 IAM 角色之类的信息。默认情况下，EC2 实例运行两个版本的 IMDS：

- 实例元数据服务版本 1 (IMDSv1) – 一种请求/响应方法
- 实例元数据服务版本 2 (IMDSv2) – 一种面向会话的方法

[imdsv2 是一种比 imdsv1 更安全的方法。](#)

默认情况下，Java SDK 首先尝试 `imdsv2` 来获取 IAM 角色，但如果失败，它将尝试 `imdsv1`。但是，由于 `imdsv1` 不太安全，因此 AWS 建议仅使用 `imdsv2`，并禁用 SDK 试用 `imdsv1`。

要使用更安全的方法，请为以下设置之一提供值为 `true`，从而禁用 SDK 使用 `imdsv1`。

- 环境变量：`AWS_EC2_METADATA_V1_DISABLED`
- JVM 系统属性：`aws.disableEc2MetadataV1`
- 共享配置文件设置：`ec2_metadata_v1_disabled`

将其中一个设置设置为 `true`，如果初始 `imdsv2` 调用失败，SDK 将不会使用 `imdsv1` 加载 IMDS 角色凭证。

使用 AWS 区域

AWS 区域 使服务客户端 AWS 服务 能够访问实际位于特定地理区域的内容。

显式配置 AWS 区域

要显式设置区域，建议您使用在 [Region](#) 类中定义的常量。这是所有公开可用区域的枚举。

要使用此类中的枚举区域创建客户端，请使用客户端生成器的 `region` 方法。

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

```
.build();
```

如果您想要使用的区域不是 `Region` 类中的枚举之一，则可以使用静态 `of` 方法创建一个新区域。借助此方法，您可以访问新区域而无需升级 SDK。

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

Note

使用构建器构建客户端后，它是不可变的，并且 AWS 区域无法更改。如果您需要 AWS 区域为同一服务使用多个客户端，则应创建多个客户端，每个区域一个。

让 SDK 根据环境自动确定区域

当您的代码在 Amazon EC2 或上运行时 AWS Lambda，您可能需要将客户端配置为使用与运行代码相同的 AWS 区域客户端。这样可以将您的代码与其运行的环境分离，并且可以更轻松地将应用程序部署到多个 AWS 区域以降低延迟或冗余。

要使用默认的凭证/区域提供程序链来根据环境确定区域，请使用客户端生成器的 `create` 方法：

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

如果您未使用该 `region` 方法明确设置，SDK 会咨询默认区域提供程序链以确定要使用的区域。AWS 区域

了解默认区域提供程序链

SDK 采用以下步骤来查找 AWS 区域：

1. 通过生成器本身使用 `region` 明确设置的所有区域优先于任何其他区域。
2. 系统会检查 `AWS_REGION` 环境变量。如果已设置该变量，将使用对应区域配置客户端。

Note

Lambda 容器设置此环境变量。

3. SDK 会检查 AWS 共享配置文件和共享凭据文件 (通常位于 `~/.aws/config` 和 `~/.aws/credentials`)。如果该 `region` 属性存在, SDK 就会使用它。
 - 如果 SDK 在两个文件中找到相同配置文件 (包括配置文件) 的 `regiondefault` 属性, 则 SDK 将使用共享凭据文件中的值。
 - `AWS_CONFIG_FILE` 环境变量可用于自定义共享配置文件的位置。
 - 可以使用 `AWS_PROFILE` 环境变量或 `aws.profile` 系统属性来指定 SDK 加载的配置文件。
4. SDK 尝试使用 Amazon EC2 实例元数据服务 (IMDS) 来确定当前正在运行的 Amazon EC2 实例的区域。
 - 为了提高安全性, 您应禁用 SDK 尝试使用 IMDS 版本 1。您可以使用与本 [the section called “安全地”](#) 节中描述的相同的设置来禁用版本 1。
5. 如果 SDK 此时仍不能确定区域, 客户端创建将失败并返回异常。

开发 AWS 应用程序时, 一种常见的方法是使用共享配置文件 (如 [凭据检索顺序](#) 所述) 来设置用于本地开发的区域, 并依靠默认的区域提供程序链来确定应用程序在 AWS 基础设施上运行时的区域。这可以明显简化客户端创建, 并保证应用程序的便携性。

查看区域的服务可用性

要查看某个区域中是否有特定 AWS 服务 内容可用, 请在服务客户端上使用 `serviceMetadata` 和 `region` 方法。

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

[有关 AWS 区域 您可以指定的内容, 请参阅 Region 类文档, 并使用服务的终端节点前缀进行查询。](#)

选择特定端点

在某些情况下 (例如在服务的预览功能正式发布之前进行测试) , 您可能需要在区域中指定特定端点。在这些情况下, 可以通过调用 `endpointOverride` 方法配置服务客户端。

例如, 要将 Amazon EC2 客户端配置为使用带有特定终端节点的欧洲 (爱尔兰) 区域, 请使用以下代码。

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

```
.build();
```

有关所有 AWS 服务的[区域及其相应终端节点](#)的当前列表，请参阅区域和终端节点。

缩短 SDK 的启动时间 AWS Lambda

的目标之一 AWS SDK for Java 2.x 是减少 AWS Lambda 函数的启动延迟。SDK 包含可缩短启动时间的更改，本主题末尾将对此进行讨论。

首先，本主题重点介绍为缩短冷启动时间可以进行的更改。这包括更改代码结构和服务客户端的配置。

使用 AWS 基于 CRT 的 HTTP 客户端

对于使用 AWS Lambda，我们建议在同步场景中使用，[AwsCrtHttpClient](#)对于异步场景，我们建议使用。

本指南中的[the section called “配置 AWS 基于 CRT 的 HTTP 客户”](#)主题描述了使用 HTTP 客户端的好处、如何添加依赖关系以及如何配置服务客户端对它们的使用。

移除未使用的 HTTP 客户端依赖项

除了明确使用 AWS 基于 CRT 的客户端外，您还可以移除 SDK 默认引入的其他 HTTP 客户端。当需要加载的库较少时，Lambda 启动时间会缩短，因此您应该移除 JVM 需要加载的所有未使用的构件。

以下 Maven pom.xml 文件片段展示了排除基于 Apache 的 HTTP 客户端和基于 Netty 的 HTTP 客户端的情况。（当您使用 AWS 基于 CRT 的客户端时，不需要这些客户端。）此示例将 HTTP 客户端项目排除在 S3 客户端依赖项之外，并添加了该aws-crt-client对象以允许访问 AWS 基于 CRT 的 HTTP 客户端。

```
<project>
  <properties>
    <aws.java.sdk.version>2.25.51</aws.java.sdk.version>
  </properties>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>${aws.java.sdk.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
</project>
```

```
        </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

将 <exclusions> 元素添加到 pom.xml 文件中的所有服务客户端依赖项中。

配置服务客户端以进行快捷查找

指定区域

创建服务客户端时，在服务客户端生成器上调用 `region` 方法。这简化了 SDK 的默认 [区域查找过程](#)，该过程会在多个位置检查 AWS 区域信息。

要使 Lambda 代码独立于区域，请在 `region` 方法中使用以下代码。此代码访问 Lambda 容器设置的 `AWS_REGION` 环境变量。

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

使用 `EnvironmentVariableCredentialProvider`

与区域信息的默认查找行为非常相似，SDK 会在多个位置查找凭证。通过在生成服务客户端时指定 [EnvironmentVariableCredentialProvider](#)，可以节省 SDK 查找过程的时间。

Note

使用此凭证提供程序可以将代码用于 Lambda 函数，但可能无法在 Amazon EC2 或其他系统上运行。

以下代码段显示了为在 Lambda 环境中使用而经过适当配置的 S3 服务客户端。

```
S3Client s3Client = S3Client.builder()

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

在 Lambda 函数处理程序之外初始化 SDK 客户端

我们建议在 Lambda 处理程序方法之外初始化 SDK 客户端。这样，如果重复使用执行上下文，则可以跳过服务客户端的初始化。通过重复使用客户端实例及其连接，处理程序方法的后续调用可更快进行。

在以下示例中，使用静态工厂方法在构造函数中初始化 S3Client 实例。如果重复使用由 Lambda 环境管理的容器，则会重复使用初始化的 S3Client 实例。

```
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.
    }
}
```


尽量减少依赖关系注入

依赖关系注入 (DI) 框架可能需要更多时间才能完成设置过程。它们可能还需要额外的依赖项，这需要一段时间才能加载。

如果需要 DI 框架，建议使用诸如 [Dagger](#) 之类的轻量级 DI 框架。

使用 Maven Archetype 瞄准 AWS Lambda

AWS Java SDK 团队开发了一个 [Maven Archetype](#) 模板，可以在最短的启动时间内启动 Lambda 项目。您可以从该原型构建 Maven 项目，并知道依赖项的配置非常适合 Lambda 环境。

要了解有关原型的更多信息并完成示例部署，请参阅此[博客文章](#)。

考虑一下适用于 Java 的 Lambda SnapStart

如果您的运行时要求兼容，则 AWS 提供适用 [SnapStart 于 Java 的 Lambda](#)。Lambda SnapStart 是一种基于基础设施的解决方案，可提高 Java 函数的启动性能。当您发布新版本的函数时，Lambda 会对其进行 SnapStart 初始化，并拍摄内存和磁盘状态的不可变加密快照。SnapStart 然后缓存快照以供重复使用。

影响启动时间的 2.x 版更改

除了您对代码所做的更改外，适用于 Java 的 SDK 2.x 版本还包括三项可缩短启动时间的主要更改：

- 使用 [jackson-jr](#)，它是一个序列化库，可以改进初始化时间
- 对日期和时间对象使用 [java.time](#) 库，此为 JDK 的一部分。
- 对记录 facade 使用 [Slf4j](#)。

其他资源

AWS Lambda 开发者指南中有一节[介绍开发非 Java 特定的 Lambda 函数的最佳实践](#)。

有关使用 Java 构建云原生应用程序的示例 AWS Lambda，请参阅此[研讨会内容](#)。该研讨会讨论了性能优化和其他最佳实践。

您可以考虑使用提前编译的静态映像来减少启动延迟。例如，您可以使用适用于 Java 的 SDK 2.x 和 Maven 来[构建 GraalVM 原生映像](#)。

HTTP客户

您可以更改用于服务HTTP客户端的客户端，也可以使用更改HTTP客户机的默认配置 AWS SDK for Java 2.x。本节讨论的HTTP客户端和设置SDK。

HTTP适用于 Java SDK 的客户端

同步客户端

SDK适用于 Java 的同步HTTP客户端实现了该[SdkHttpClient](#)接口。同步服务客户端（例如S3Client或）需要使用同步HTTP客户端。DynamoDbClient AWS SDK for Java 提供了三个同步HTTP客户端。

ApacheHttpClient（默认）

[ApacheHttpClient](#)是同步服务HTTP客户端的默认客户端。有关配置 ApacheHttpClient 的信息，请参阅[配置基于 Apache 的 HTTP 客户端](#)。

AwsCrtHttpClient

[AwsCrtHttpClient](#)提供高吞吐量和无阻塞 IO。它建立在 AWS 通用运行时 (CRT) Http 客户端之上。有关配置 AwsCrtHttpClient 并将其用于服务客户端的信息，请参阅[the section called “配置 AWS 基于 CRT 的 HTTP 客户”](#)。

URLConnectionHttpClient

要最大限度地减少应用程序使用的 jar 和第三方库的数量，可以使用。[URLConnectionHttpClient](#)有关配置 UrlConnectionHttpClient 的信息，请参阅[配置基于 URLConnection 的 HTTP 客户端](#)。

异步客户端

SDK适用于 Java 的异步HTTP客户端实现了该[SdkAsyncHttpClient](#)接口。异步服务客户端（例如S3AsyncClient或）需要使用异步HTTP客户端。DynamoDbAsyncClient AWS SDK for Java 提供了两个异步HTTP客户端。

NettyNioAsyncHttpClient（默认）

[NettyNioAsyncHttpClient](#)是异步HTTP客户端使用的默认客户端。有关配置 NettyNioAsyncHttpClient 的信息，请参阅[the section called “配置基于 Netty 的 HTTP 客户端”](#)。

AwsCrtAsyncHttpClient

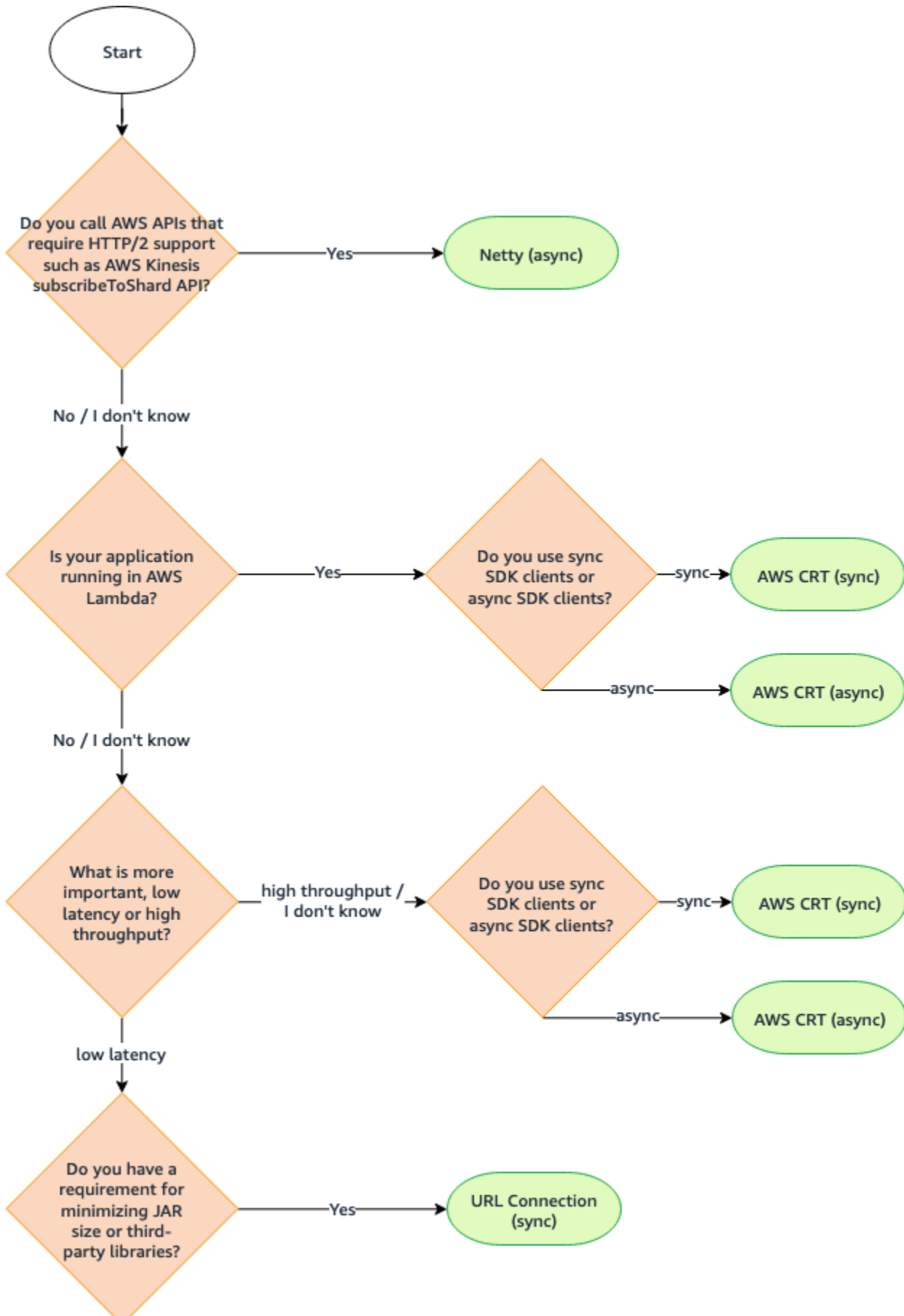
[AwsCrtAsyncHttpClient](#) 基于 AWS 通用运行时 (CRT) HTTP 客户端。有关配置 `AwsCrtAsyncHttpClient` 的信息，请参阅 [the section called “配置 AWS 基于 CRT 的 HTTP 客户端”](#)。

HTTP 客户推荐

选择 HTTP 客户端实现时，有几个因素会起作用。以下信息可帮助您做出决定。

建议流程图

以下流程图提供了一般指导，可帮助您确定要使用哪个 HTTP 客户端。



HTTP客户比较

下表提供了每个HTTP客户端的详细信息。

HTTP客户	同步还是异步	何时使用	限制/缺点
基于 Apache 的客户端 HTTP (默认同步HTTP客户端)	同步	如果您更需要低延迟而不是高吞吐量，请使用它	与其他HTTP客户端相比，启动时间较慢
URLConnection 基于的HTTP客户端	同步	如果您限制第三方依赖项有硬性要求，请使用它	不支持该HTTPATCH方法，这是某些操作所必需的，APIS 例如 Amazon APIGateway Update 操作
AWS CRT基于同步的HTTP客户端 ¹	同步	<ul style="list-style-type: none"> 如果您的应用程序正在中运行，请使用它 AWS Lambda 如果您更需要高吞吐量而不是低延迟，请使用它 如果您更喜欢同步SDK客户端，请使用它 	不支持以下 Java 系统属性： <ul style="list-style-type: none"> javax.net.ssl。keyStore javax.net.ssl。keyStoreP assword javax.net.ssl。trustStore javax.net .ssl。trustStor ePassword
基于 Netty 的客户端 HTTP	异步	<ul style="list-style-type: none"> 如果您的应用程序调用需要 HTTP /2 支持 (例如 Kinesis) APIs，请使用它 API SubscribeToShard 	与其他HTTP客户端相比，启动时间较慢

HTTP客户	同步还是异步	何时使用	限制/缺点
(默认异步HTTP客户端)			
AWS CRT ^{基于异步} HTTP客户端 ¹	异步	<ul style="list-style-type: none"> 如果您的应用程序在 AWS Lambda 中运行，请使用它 如果您更需要高吞吐量而不是低延迟，请使用它 如果您更喜欢异步SDK客户端，请使用它 	<ul style="list-style-type: none"> 不支持需要 HTTP /2 支持的服务客户端，例如KinesisAsyncClient 和 TranscribeStreamingAsyncClient 不支持以下 Java 系统属性： <ul style="list-style-type: none"> javax.net.ssl.keyStore javax.net.ssl.keyStorePassword javax.net.ssl.trustStore javax.net.ssl.trustStorePassword

¹ 由于 AWS CRT基于这些客户端的额外好处，我们建议您尽可能使用这些HTTP客户端。

智能配置默认值

AWS SDK for Java 2.x (版本 2.17.102 或更高版本) 提供智能配置默认功能。此功能优化了两个 HTTP客户端属性以及其他不影响HTTP客户端的属性。

智能配置默认值会根据您提供的默认模式值，为 `connectTimeoutInMillis` 和 `tlsNegotiationTimeoutInMillis` 属性设置合理的值。您可以根据应用程序的特性选择默认模式值。

有关智能配置默认值以及如何选择最适合您的应用程序的默认模式值的更多信息，请参阅[AWS SDKs 和工具参考指南](#)。

以下是为应用程序设置默认模式的四种方法。

Service client

使用服务客户端生成器直接在服务客户端上配置默认模式。以下示例将 `DynamoDbClient` 的默认模式设置为 `auto`。

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

System property

您可以使用 `aws.defaultsMode` 系统属性来指定默认模式。如果在 Java 中设置系统属性，则需要在初始化任何服务客户端之前设置该属性。

以下示例演示了如何使用在 Java 中设置的系统属性将默认模式设置为 `auto`。

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

以下示例演示了如何使用 `java` 命令的 `-D` 选项将默认模式设置为 `auto`。

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

Environment variable

为环境变量 `AWS_DEFAULTS_MODE` 设置一个值以选择应用程序的默认模式。

以下信息显示了使用环境变量将默认模式的值设置为 `auto` 时，所运行的命令。

操作系统	设置环境变量的命令
Linux、macOS 或 Unix	<code>export AWS_DEFAULTS_MODE=auto</code>

操作系统	设置环境变量的命令
Windows	set AWS_DEFAULTS_MODE=auto

AWS config file

您可以向共享 AWS config 文件添加 `defaults_mode` 配置属性，如下例所示。

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

如果您使用系统属性、环境变量或 AWS 配置文件全局设置默认模式，则可以在构建 HTTP 客户端时覆盖这些设置。

使用该 `httpClientBuilder()` 方法构建 HTTP 客户端时，设置仅适用于您正在构建的实例。[此处](#) 显示了此方法一个示例。本示例中基于 Netty 的 HTTP 客户端会覆盖为和全局设置的任何默认模式值。`connectTimeoutInMillis` `tlsNegotiationTimeoutInMillis`

配置基于 Apache 的 HTTP 客户端

默认情况下，中的同步服务客户端 AWS SDK for Java 2.x 使用基于 Apache 的 HTTP 客户端。[ApacheHttpClient](#) 该软件开发工具包基 `ApacheHttpClient` 于 Apache [HttpClient](#)。

SDK 还提供 [URLConnectionHttpClient](#)，加载速度更快，但功能较少。有关配置 `URLConnectionHttpClient` 的信息，请参阅 [the section called “配置基于 URLConnection 的 HTTP 客户端”](#)。

要查看可供您使用的全套配置选项，请参阅 [ApacheHttpClient.Builder](#) 和 [ProxyConfiguration.Builder](#)。`ApacheHttpClient`

访问 `ApacheHttpClient`

在大多数情况下，您无需进行任何显式配置即可使用 `ApacheHttpClient`。您只需声明您的服务客户端，SDK 将使用标准值为您配置 `ApacheHttpClient`。

如果要显式配置 `ApacheHttpClient` 或将其用于多个服务客户端，则需要将其设置为可供配置。

无需配置

当您在 Maven 中声明对服务客户端的依赖项时，SDK 会添加对 `apache-client` 构件的运行时系统依赖项。这使得 `ApacheHttpClient` 类在运行时可供您的代码使用，但在编译时不可用。如果您没有配置基于 Apache 的 HTTP 客户端，则无需为其指定依赖项。

在以下 Maven `pom.xml` 文件的 XML 片段中，使用 `<artifactId>s3</artifactId>` 声明的依赖项会自动引入基于 Apache 的 HTTP 客户端。您无需专门为其声明依赖项。

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

有了这些依赖项，您无法进行任何显式 HTTP 配置更改，因为 `ApacheHttpClient` 库仅位于运行时系统类路径上。

需要配置

要配置 `ApacheHttpClient`，您需要在编译时添加对 `apache-client` 库的依赖项。

请参阅以下 Maven `pom.xml` 文件示例来配置 `ApacheHttpClient`。

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
```

```
        <artifactId>bom</artifactId>
        <version>2.17.290</version>
        <type>pom</type>
        <scope>import</scope>
    </dependency>
</dependencies>
</dependencyManagement>
<dependencies>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>s3</artifactId>
    </dependency>
    <!-- By adding the apache-client dependency, ApacheHttpClient will be added to
         the compile classpath so you can configure it. -->
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>apache-client</artifactId>
    </dependency>
</dependencies>
```

使用和配置 `ApacheHttpClient`

您可以在生成服务客户端的同时配置一个 `ApacheHttpClient` 实例，也可以将单个实例配置为在多个服务客户端之间共享。

无论采用哪种方法，都可以使用 [ApacheHttpClient.Builder](#) 来配置基于 Apache 的 HTTP 客户端的属性。

最佳实践： 将一个 `ApacheHttpClient` 实例专用于一个服务客户端

如果您需要配置 `ApacheHttpClient` 实例，建议您生成专用 `ApacheHttpClient` 实例。您可以通过使用服务客户端生成器的 `httpClientBuilder` 方法来执行此操作。这样，HTTP 客户端的生命周期就由 SDK 管理，这有助于避免在不再需要 `ApacheHttpClient` 实例却不关闭实例时可能发生的内存泄漏。

以下示例创建了一个 `S3Client` 并配置了具有 `maxConnections` 和 `connectionTimeout` 值的 `ApacheHttpClient` 嵌入式实例。HTTP 实例是使用 `S3Client.Builder` 的 `httpClientBuilder` 方法创建的。

导入

```
import software.amazon.awssdk.http.apache.ApacheHttpClient;
```

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

代码

```
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.
```

替代方法：共享 **ApacheHttpClient** 实例

为了帮助降低应用程序的资源 and 内存使用量，您可以配置 `ApacheHttpClient` 并在多个服务客户端之间共享该客户端。将共享 HTTP 连接池，从而降低资源使用量。

Note

共享 `ApacheHttpClient` 实例时，必须在准备好弃置实例时将其关闭。服务客户端关闭后，SDK 不会关闭实例。

以下示例配置了一个基于 Apache 的 HTTP 客户端，该客户端由两个服务客户端使用。配置的 `ApacheHttpClient` 实例将传递给每个生成器的 `httpClient` 方法。当不再需要服务客户端和 HTTP 客户端时，代码会显式关闭它们。代码最后关闭 HTTP 客户端。

导入

```
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;
```

代码

```
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.
```

代理配置示例

以下代码段使用了[适用于 Apache HTTP 客户端的代理配置生成器](#)。

```
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();
```

以下命令行片段显示了代理配置的等效 Java 系统属性。

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

使用环境变量的等效设置为：

```
// 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 客户端目前不支持 HTTPS 代理系统属性或 HTTPS_PROXY 环境变量。

配置基于 URLConnection 的 HTTP 客户端

与默认客户端相比，AWS SDK for Java 2.x 它提供了更轻量 [URLConnectionHttpClient](#) 的 HTTP 客户端。ApacheHttpClientURLConnectionHttpClient 基于 Java 的 [URLConnection](#)。

URLConnectionHttpClient 的加载速度比基于 Apache 的 HTTP 客户端快，但功能较少。由于加载速度更快，因此该客户端是 Java AWS Lambda 函数的 [良好解决方案](#)。

URLConnectionHttpClient 提供了几个 [可配置的选项](#) 供您使用。

Note

URLConnectionHttpClient 不支持 HTTP PATCH 方法。
少数 AWS API 操作需要补丁请求。这些操作的名称通常以 Update* 开头。以下是几个示例。

- AWS Security Hub API 中的 @@ [几个 Update* 操作](#) 以及 [BatchUpdateFindings](#) 操作
- 所有 Amazon API Gateway API 的 [Update* 操作](#)
- 亚马逊 WorkDocs API 中的 @@ [几项 Update* 操作](#)

如果你可能使用 UrlConnectionHttpClient，请先参阅你正在使用 AWS 服务的 API 参考。了解您需要的操作是否使用 PATCH 操作。

访问 `URLConnectionHttpClient`

要配置和使用 `URLConnectionHttpClient`，请在 `pom.xml` 文件中声明对 `url-connection-client` Maven 构件的依赖项。

与 `ApacheHttpClient` 不同的是，`URLConnectionHttpClient` 不会自动添加到您的项目中，因此，要使用该客户端必须对其进行特别声明。

以下 `pom.xml` 文件示例显示了使用和配置 HTTP 客户端所需的依赖项。

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

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

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

使用和配置 `URLConnectionHttpClient`

您可以在生成服务客户端的同时配置一个 `URLConnectionHttpClient` 实例，也可以将单个实例配置为在多个服务客户端之间共享。

无论采用哪种方法，您都可以使用 [URLConnectionHttpClient.Builder](#) 来配置基于 `URLConnection` 的 HTTP 客户端的属性。

最佳实践： 将一个 `URLConnectionHttpClient` 实例专用于一个服务客户端

如果您需要配置 `URLConnectionHttpClient` 实例，建议您生成专用 `URLConnectionHttpClient` 实例。您可以通过使用服务客户端生成器的 `httpClientBuilder`

方法来执行此操作。这样，HTTP 客户端的生命周期就由 SDK 管理，这有助于避免在不再需要 `URLConnectionHttpClient` 实例却不关闭实例时可能发生的内存泄漏。

以下示例创建了一个 `S3Client` 并配置了具有 `socketTimeout` 和 `proxyConfiguration` 值的 `URLConnectionHttpClient` 嵌入式实例。`proxyConfiguration` 方法采用类型为 `Consumer<ProxyConfiguration.Builder>` 的 Java lambda 表达式。

导入

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

代码

```
// 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.
```

替代方法：共享 `URLConnectionHttpClient` 实例

为了帮助降低应用程序的资源 and 内存使用量，您可以配置 `URLConnectionHttpClient` 并在多个服务客户端之间共享该客户端。将共享 HTTP 连接池，从而降低资源使用量。

Note

共享 `URLConnectionHttpClient` 实例时，必须在准备好弃置实例时将其关闭。服务客户端关闭后，SDK 不会关闭实例。

以下示例配置了一个基于 `URLConnection` 的 HTTP 客户端，该客户端由两个服务客户端使用。配置的 `URLConnectionHttpClient` 实例将传递给每个生成器的 `httpClient` 方法。当不再需要服务客户端和 HTTP 客户端时，代码会显式关闭它们。代码最后关闭 HTTP 客户端。

导入

```
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;
```

代码

```
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();
```


将 `URLConnectionHttpClient` 和 `ApacheHttpClient` 一起使用

在应用程序中使用 `URLConnectionHttpClient` 时，您必须使用服务客户端生成器的 `httpClientBuilder` 方法为每个服务客户端提供一个 `URLConnectionHttpClient` 实例或一个 `ApacheHttpClient` 实例。

如果程序使用多个服务客户端，并且同时存在以下两个条件，则会发生异常：

- 一个服务客户端配置为使用一个 `URLConnectionHttpClient` 实例
- 另一个服务客户端使用默认 `ApacheHttpClient`，但没有使用 `httpClient()` 或 `httpClientBuilder()` 方法显式生成

该异常将声明在类路径中找到了多个 HTTP 实现。

以下示例代码片段会导致异常。

```
// 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();
```

通过显式使用 `ApacheHttpClient` 来配置 `S3Client`，可避免异常。

```
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

要显式创建 `ApacheHttpClient`，必须在 Maven 项目文件中[添加对 `apache-client` 构件的依赖项](#)。

代理配置示例

以下代码片段将[代理配置生成器用于 URL 连接 HTTP 客户端](#)。

```
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();
```

以下命令行片段显示了代理配置的等效 Java 系统属性。

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

使用环境变量的等效设置为：

```
// 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

基于 `URLConnection` 的 HTTP 客户端目前不支持 HTTPS 代理系统属性或 `HTTPS_PROXY` 环境变量。

配置基于 Netty 的 HTTP 客户端

中用于异步操作的默认 HTTP 客户端 AWS SDK for Java 2.x 是基于 Netty 的。[NettyNioAsyncHttpClient](#) 基于 Netty 的客户端以 [Netty 项目](#) 的异步事件驱动网络框架为基础。

作为备选 HTTP 客户端，您可以使用新的[基于 AWS CRT 的 HTTP 客户端](#)。本主题演示如何配置 `NettyNioAsyncHttpClient`。

访问 `NettyNioAsyncHttpClient`

在大多数情况下，您无需在异步程序中进行任何显式配置即可使用 `NettyNioAsyncHttpClient`。您只需声明您的异步客户端，SDK 将使用标准值为您配置 `NettyNioAsyncHttpClient`。

如果要显式配置 `NettyNioAsyncHttpClient` 或将其用于多个服务客户端，则需要将其设置为可供配置。

无需配置

当您在 Maven 中声明对服务客户端的依赖项时，SDK 会添加对 `netty-nio-client` 构件的运行时系统依赖项。这使得 `NettyNioAsyncHttpClient` 类在运行时可供您的代码使用，但在编译时不可用。如果您没有配置基于 Netty 的 HTTP 客户端，则无需为其指定依赖项。

在以下 Maven `pom.xml` 文件的 XML 片段中，使用 `<artifactId>dynamodb-enhanced</artifactId>` 声明的依赖项会以传递的方式引入基于 Netty 的 HTTP 客户端。您无需专门为其声明依赖项。

```
<dependencyManagement>
  <dependencies>
    <dependency>
```

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

有了这些依赖项，您无法进行任何 HTTP 配置更改，因为 NettyNioAsyncHttpClient 库仅位于运行时系统类路径上。

需要配置

要配置 NettyNioAsyncHttpClient，您需要在编译时添加对 netty-nio-client 构件的依赖项。

请参阅以下 Maven pom.xml 文件示例来配置 NettyNioAsyncHttpClient。

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>bom</artifactId>
            <version>2.17.290</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
<dependencies>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>dynamodb-enhanced</artifactId>
    </dependency>
    <!-- By adding the netty-nio-client dependency, NettyNioAsyncHttpClient will
be
```

```
        added to the compile classpath so you can configure it. -->
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>netty-nio-client</artifactId>
    </dependency>
</dependencies>
```

使用和配置 `NettyNioAsyncHttpClient`

您可以在生成服务客户端的同时配置一个 `NettyNioAsyncHttpClient` 实例，也可以将单个实例配置为在多个服务客户端之间共享。

无论采用哪种方法，您都可以使用 [NettyNioAsyncHttpClient.Builder](#) 来配置基于 Netty 的 HTTP 客户端实例的属性。

最佳实践： 将一个 `NettyNioAsyncHttpClient` 实例专用于一个服务客户端

如果您需要配置 `NettyNioAsyncHttpClient` 实例，建议您生成专用 `NettyNioAsyncHttpClient` 实例。您可以通过使用服务客户端生成器的 `httpClientBuilder` 方法来执行此操作。这样，HTTP 客户端的生命周期就由 SDK 管理，这有助于避免在不再需要 `NettyNioAsyncHttpClient` 实例却不关闭实例时可能发生的内存泄漏。

以下示例创建了一个 `DynamoDbAsyncClient` 实例，该实例供 `DynamoDbEnhancedAsyncClient` 实例使用。该 `DynamoDbAsyncClient` 实例包含具有 `connectionTimeout` 和 `maxConcurrency` 值的 `NettyNioAsyncHttpClient` 实例。HTTP 实例是使用 `DynamoDbAsyncClient.Builder` 的 `httpClientBuilder` 方法创建的。

导入

```
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;
```

代码

```
// 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();

```

替代方法：共享 **NettyNioAsyncHttpClient** 实例

为了帮助降低应用程序的资源 and 内存使用量，您可以配置 `NettyNioAsyncHttpClient` 并在多个服务客户端之间共享该客户端。将共享 HTTP 连接池，从而降低资源使用量。

Note

共享 `NettyNioAsyncHttpClient` 实例时，必须在准备好弃置实例时将其关闭。服务客户端关闭后，SDK 不会关闭实例。

以下示例配置了一个基于 Netty 的 HTTP 客户端，该客户端由两个服务客户端使用。配置的 `NettyNioAsyncHttpClient` 实例将传递给每个生成器的 `httpClient` 方法。当不再需要服务客户端和 HTTP 客户端时，代码会显式关闭它们。代码最后关闭 HTTP 客户端。

导入

```

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;
```

代码

```
// 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.
```

代理配置示例

以下代码段使用了[适用于 Netty HTTP 客户端的代理配置生成器](#)。

```
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();
```

以下命令行片段显示了代理配置的等效 Java 系统属性。

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

Important

要使用任何 HTTPS 代理系统属性，必须在代码中将 `scheme` 属性设置为 `https`。如果未在代码中设置 `scheme` 属性，则架构默认为 HTTP，SDK 仅查找 `http.*` 系统属性。

使用环境变量的等效设置为：

```
// 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
```


配置 AWS 基于 CRT 的 HTTP 客户

AWS 基于 CRT 的 HTTP 客户端包括同步客户端[AwsCrtHttpClient](#)和异步[AwsCrtAsyncHttpClient](#)客户端。AWS 基于 CRT 的 HTTP 客户端具有以下 HTTP 客户端优势：

- 更快的 SDK 启动时间
- 更小的内存占用空间
- 降低的延迟时间
- 连接运行状况管理
- DNS 负载均衡

AWS SDK 中基于 CRT 的组件

本主题中介绍 AWS 的基于 CRT 的 HTTP 客户端和 AWS 基于 CRT 的 S3 客户端是软件开发工具包中的不同组件。

同步和异步基于 AWS CRT 的 HTTP 客户端是 SDK HTTP 客户端接口的实现，用于一般 HTTP 通信。它们是 SDK 中其他同步或异步 HTTP 客户端的替代方案，提供额外优点。

[AWS 基于 CRT 的 S3 客户端](#)是 [S3 AsyncClient](#) 接口的实现，用于与 Amazon S3 服务配合使用。它是基于 Java 的 [S3AsyncClient](#) 接口实现的替代方案，具有多种优点。

尽管两个组件都使用[AWS 公共运行时](#)中的库，但 AWS 基于 CRT 的 HTTP 客户端不使用 [aws-c-s3 库](#)，也不支持 [S3 分段上传 API](#) 功能。相比之下，AWS 基于 CRT 的 S3 客户端是专门为支持 S3 分段上传 API 功能而设计的。

访问 AWS 基于 CRT 的 HTTP 客户端

在使用 AWS 基于 CRT 的 HTTP 客户端之前，请将最低版本为 2.22.0 的 `aws-crt-client` 工件添加到项目的依赖项中。

以下 Maven `pom.xml` 显示了使用物料清单 (BOM) 机制声明的 AWS 基于 CRT 的 HTTP 客户端。

```
<project>
  <properties>
    <aws.sdk.version>2.22.0</aws.sdk.version>
  </properties>
  <dependencyManagement>
    <dependencies>
      <dependency>
```

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

访问 Maven Central 存储库以获取[最新版本](#)。

使用和配置 AWS 基于 CRT 的 HTTP 客户端

您可以在构建服务客户端的同时配置 AWS 基于 CRT 的 HTTP 客户端，也可以将单个实例配置为在多个服务客户端之间共享。

无论采用哪种方法，您都可以使用生成器来[配置 AWS 基于 CRT 的 HTTP 客户端实例的属性](#)。

最佳实践： 将一个实例专用于一个服务客户端

如果您需要配置 AWS 基于 CRT 的 HTTP 客户端的实例，我们建议您将该实例与服务客户端一起构建，从而将其专用。您可以通过使用服务客户端生成器的 `httpClientBuilder` 方法来执行此操作。这样，HTTP 客户端的生命周期就由 SDK 管理，这有助于避免在不再需要 AWS 基于 CRT 的 HTTP 客户端实例时未关闭时可能发生的内存泄漏。

以下示例创建一个 S3 服务客户端，并使用和值配置 AWS 基于 CRT 的 HTTP 客户端 `connectionTimeout`。 `maxConcurrency`

Synchronous client

导入

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

代码

```
// 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

导入

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

代码

```
// 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();
```

替代方法：共享实例

为了帮助降低应用程序的资源 and 内存使用量，您可以配置 AWS 基于 CRT 的 HTTP 客户端，并在多个服务客户端之间共享该客户端。将共享 HTTP 连接池，从而降低资源使用量。

Note

共享 AWS 基于 CRT 的 HTTP 客户端实例时，您必须在准备好处置时将其关闭。服务客户端关闭后，SDK 不会关闭实例。

以下示例使用 `connectionTimeout` 和 `maxConcurrency` 值配置 AWS 基于 CRT 的 HTTP 客户端实例。配置的实例将传递给每个服务客户端的生成器的 `httpClient` 方法。当不再需要服务客户端和 HTTP 客户端时，它们将被显式关闭。HTTP 客户端最后关闭。

Synchronous client

导入

```
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;
```

代码

```
// 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

导入

```
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;
```

代码

```
// 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.
```

将 AWS 基于 CRT 的 HTTP 客户端设置为默认客户端

你可以设置你的 Maven 编译文件，让 SDK 使用 AWS 基于 CRT 的 HTTP 客户端作为服务客户端的默认 HTTP 客户端。

为此，您可以向每个服务客户端构件添加一个具有默认 HTTP 客户端依赖项的 `exclusions` 元素。

在以下 `pom.xml` 示例中，软件开发工具包将 AWS 基于 CRT 的 HTTP 客户端用于 S3 服务。如果您的代码中的服务客户端是 `S3AsyncClient`，则 SDK 使用 `AwsCrtAsyncHttpClient`。如果服务客户端是 `S3Client`，则 SDK 使用 `AwsCrtHttpClient`。在此设置下，默认的基于 Netty 的异步 HTTP 客户端和默认的基于 Apache 的同步 HTTP 不可用。

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

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>aws-crt-client</artifactId>
</dependency>
</dependencies>
</project>
```

请访问 [Maven Central 存储库](#) 以获取最新的 [VERSION](#) 值。

Note

如果在一个 pom.xml 文件中声明了多个服务客户端，则所有服务客户端都需要 `exclusions` XML 元素。

使用 Java 系统属性

要使用 AWS 基于 CRT 的 HTTP 客户端作为应用程序的默认 HTTP，可以将 Java 系统属性的 `software.amazon.awssdk.http.async.service.impl` 设置为 `software.amazon.awssdk.http.crt.AwsCrtSdkHttpService`。

要在应用程序启动期间设置，请运行类似以下示例的命令。

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

然后使用以下代码段在应用程序代码中设置系统属性。

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

Note

当你使用系统属性配置基于 AWS CRT 的 HTTP 客户端的使用时，你需要在 pom.xml 文件中添加对 `aws-crt-client` 工件的依赖关系。

基于 AWS CRT 的 HTTP 客户端的高级配置

您可以使用 AWS 基于 CRT 的 HTTP 客户端的各种配置设置，包括连接运行状况配置和最大空闲时间。您可以查看适用于 `AwsCrtAsyncHttpClient` 的[可用配置选项](#)。您可以为 `AwsCrtHttpClient` 配置相同的选项。

连接运行状况配置

您可以使用 HTTP 客户端生成器上的 `connectionHealthConfiguration` 方法为 AWS 基于 CRT 的 HTTP 客户端配置连接运行状况配置。

以下示例创建了一个 S3 服务客户端，该客户端使用 AWS 基于 CRT 的 HTTP 客户端实例，该实例配置了连接运行状况配置和连接的最大空闲时间。

Synchronous client

导入

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

代码

```
// 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

导入


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

代码

```
// 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 支持

AWS 基于 CRT 的 HTTP 客户端尚不支持 HTTP/2 协议，但计划在将来的版本中推出。

同时，如果您使用的是需要 HTTP/2 支持的服务客户端，例如[KinesisAsyncClient](#)或[TranscribeStreamingAsyncClient](#)，请考虑改用。[NettyNioAsyncHttpClient](#)

代理配置示例

以下代码段显示了如何使用 [ProxyConfiguration.Builder](#) 在代码中配置代理设置。

Synchronous client

导入

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

代码

```
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

导入

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

代码

```
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();
```

以下命令行片段显示了代理配置的等效 Java 系统属性。

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

⚠ Important

要使用任何 HTTPS 代理系统属性，必须在代码中将 `scheme` 属性设置为 `https`。如果未在代码中设置 `scheme` 属性，则架构默认为 HTTP，SDK 仅查找 `http.*` 系统属性。

使用环境变量的等效设置为：

```
// 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
```

配置 HTTP 代理

您可以使用代码、设置 Java 系统属性或设置环境变量来配置 HTTP 代理。

在代码中配置

在生成服务客户端时，您可以使用特定于客户端的 `ProxyConfiguration` 生成器在代码中配置代理。以下代码显示了 Amazon S3 服务客户端使用的基于 Apache 的 HTTP 客户端的代理配置示例。

```
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();
```

本主题中提及的每个 HTTP 客户端都在其详细介绍部分提供了一个代理配置示例。

- [阿帕奇 HTTP 客户端](#)
- [基于 URLConnection 的 HTTP 客户端](#)
- [基于 Netty 的 HTTP 客户端](#)
- [AWS 基于 CRT 的 HTTP](#)

使用外部设置配置 HTTP 代理

即使您没有在代码中明确使用 ProxyConfiguration 构建器，SDK 也会查找外部设置来配置默认代理配置。

默认情况下，SDK 首先搜索 JVM 系统属性。即使找到一个属性，SDK 也会使用该值和任何其他系统属性值。如果没有可用的系统属性，SDK 会查找代理环境变量。

SDK 可以使用以下 Java 系统属性和环境变量。

Java 系统属性

系统属性	描述	HTTP 客户端支持
http.proxyHost	HTTP 代理服务器主机名	全部
http.proxyPort	HTTP 代理服务器端口号	全部
http.proxyUser	HTTP 代理身份验证用户名	全部
http.proxyPassword	HTTP 代理身份验证密码	全部
http.nonProxyHosts	应绕过代理直接访问的主机列表。 使用 HTTPS 时，此列表也是有效的。	全部
https.proxyHost	HTTPS 代理服务器主机名	Netty、CRT
https.proxyPort	HTTPS 代理服务器端口号	Netty、CRT

系统属性	描述	HTTP 客户端支持
https.proxyUser	HTTPS 代理身份验证用户名	Netty、CRT
https.proxyPassword	HTTPS 代理身份验证密码	Netty、CRT

环境变量

环境变量	描述	HTTP 客户端支持
HTTP_PROXY ¹	采用 HTTP 架构的有效网址	全部
HTTPS_PROXY ¹	采用 HTTPS 架构的有效网址	Netty、CRT
NO_PROXY ²	应绕过代理直接访问的主机列表。该列表对于 HTTP 和 HTTPS 均有效。	全部

查看密钥和脚注

全部-SDK 提供的所有 HTTP 客户端— `URLConnectionHttpClient`

`ApacheHttpClient`、`NettyNioAsyncHttpClient`、`AwsCrtAsyncHttpClient`。

Netty-基于 Netty 的 HTTP 客户端 (`NettyNioAsyncHttpClient`)。

CRT-AWS 基于 CRT 的 HTTP 客户端 (`AwsCrtHttpClient` 和 `AwsCrtAsyncHttpClient`)。

¹ 查询的环境变量 (`HTTP_PROXY` 是否 `HTTPS_PROXY` 为) 取决于客户端中的方案设

置。`ProxyConfiguration` 默认方案是 HTTP。以下代码段显示了如何将用于环境变量解析的方案更改为 HTTPS。

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

² `NO_PROXY` 环境变量支持在主机名之间混用 “|” 和 “,” 分隔符。主机名可能包含 “*” 通配符。

使用多种设置组合

您可以在代码、系统属性和环境变量中组合使用 HTTP 代理设置。

Example — 由系统属性和代码提供的配置

```
// 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();
```

SDK 解析了以下代理设置。

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

Example — 系统属性和环境变量都可用

每个 HTTP 客户端的ProxyConfiguration生成器都提供名为useSystemPropertyValues和的设置useEnvironmentVariablesValues。默认情况下，这两个设置都是true。在这种情况下true，SDK 会自动使用系统属性或环境变量中的值作为ProxyConfiguration构建器未提供的选项。

⚠ Important

系统属性的优先级高于环境变量。如果找到 HTTP 代理系统属性，SDK 将从系统属性中检索所有值，而不会从环境变量中检索任何值。如果要将环境变量置于系统属性之上，`useSystemPropertyValues` 请将设置为 `false`。

在本示例中，运行时可以使用以下设置：

```
// 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"
```

使用以下语句之一创建服务客户端。这些语句都没有明确设置代理设置。

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

SDK 解析了以下代理设置：

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

由于服务客户端具有默认代理设置，因此 SDK 会搜索系统属性，然后搜索环境变量。由于系统属性设置优先于环境变量，因此 SDK 仅使用系统属性。

如果将系统属性的使用更改为 `false` 为如以下代码所示，则 SDK 仅解析环境变量。

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

使用 HTTP 解析的代理设置有：

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

AWS SDK for Java 2.x 的异常处理

要使用开发工具包构建高质量的应用程序，必须了解 AWS SDK for Java 2.x 在什么情况下会引发异常以及它以什么方式引发异常。接下来几节介绍开发工具包引发异常的几种不同情况，以及如何正确地处理这些异常。

为什么使用未选中的异常？

出于以下原因，AWS SDK for Java 使用运行时（或未选中的）异常而不是选中的异常：

- 使开发人员能够精细控制要处理哪些错误，而不是必须处理无关紧要的异常情况（这会导致代码极其冗长）
- 避免大型应用程序因使用选中的异常而固有的可扩展性问题

一般来说，小型应用程序使用选中的异常是可以的，但随着应用程序的大小和复杂程度增加，这样做就会出现一些问题。

AwsServiceException（和子类）

[AwsServiceException](#) 是您在使用时遇到的最常见的异常 AWS SDK for Java。

[AwsServiceException](#) 是更通 [SdkServiceException](#) 用的子类。AwsServiceExceptions 表示来

自 a 的错误响应AWS 服务。例如，如果您尝试终止不存在的 Amazon EC2 实例，Amazon EC2 会返回错误响应，而且引发的 `AwsServiceException` 中会包含该错误响应的所有详细信息。

当您遇到 `AwsServiceException` 时，您就会知道，您的请求已成功发送到 AWS 服务，但无法成功处理。这可能是由于请求的参数中存在错误，或者是因为服务端的问题。

`AwsServiceException` 为您提供很多信息，例如：

- 返回的 HTTP 状态代码
- 返回的 AWS 错误代码
- [AwsErrorDetails](#)课堂上来自服务的详细错误消息
- 已失败请求的 AWS 请求 ID

在某些情况下，会引发 `AwsServiceException` 的一个特定于服务的子类，使开发人员能够通过捕获模块精细控制如何处理错误情况。的 Java SDK API 参考中[AwsServiceException](#)显示了大量的 `AwsServiceException` 子类。使用子类链接可深入查看服务引发的细粒度异常。

例如，以下指向 SDK API 参考的链接中提供了一些常用 AWS 服务的异常层次结构。每个页面上的子类列表显示了您的代码可以捕获的特定异常。

- [Amazon S3](#)
- [DynamoDB](#)
- [Amazon SQS](#)

要了解有关异常的更多信息，请检查[AwsErrorDetails](#)对象 `errorCode` 上的。您可以使用该 `errorCode` 值在服务指南 API 中查找信息。例如，如果捕获到 `S3Exception` 且 `AwsErrorDetails#errorCode()` 的值为 `InvalidRequest`，则使用《Amazon S3 API Reference》中的 [List of error codes](#) 来查看更多详细信息。

SdkClientException

[SdkClientException](#) 表示 Java 客户端代码内部出现问题，无论是在尝试向发送请求时AWS还是尝试解析来自AWS的响应时。在一般情况下，`SdkClientException` 比 `SdkServiceException` 严重，前者指示出现严重问题，导致客户端无法对 AWS 服务进行服务调用。例如，如果您在尝试对一个客户端执行操作时网络连接不可用，AWS SDK for Java会引发 `SdkClientException`。

异常和重试行为

适用于 Java 的 SDK 会针对多个[客户端异常](#)以及从 AWS 服务响应中收到的[HTTP 状态代码](#)重试请求。这些错误作为服务客户端默认使用的旧版 `RetryMode` 的一部分进行处理。有关 [RetryMode](#) 的 Java API 参考描述了可用于配置模式的各种方式。

要自定义触发自动重试的异常和 HTTP 状态代码，请使用添加 [RetryOnExceptionsCondition](#) 和 [RetryOnStatusCodeCondition](#) 实例的 [RetryPolicy](#) 来配置服务客户端。

重试

偶尔会由于意想不到的原因调用失败。AWS 服务 如果重试呼叫，某些错误，例如限制（超出速率）或暂时错误，可能会成功。AWS SDK for Java 2.x 具有检测此类错误并自动重试所有客户端默认启用的呼叫的内置机制。

本页介绍其工作原理、如何配置不同的模式以及如何定制重试行为。

重试策略

重试策略是中用来实现重试SDK的机制。每个SDK客户端都有在构建时创建的重试策略，该策略在生成客户端后无法修改。

重试策略具有以下职责。

- 将异常归类为可重试或不可重试。
- 计算下次尝试之前要等待的建议延迟。
- 维护一个[令牌存储桶](#)，[该存储桶](#)提供了一种在很大一部分请求失败且重试失败时停止重试的机制。

Note

在 2.26.0 版本发布重试策略之前 SDK，重试策略在中提供了重试机制。SDK 重试策略 API 由软件包中的核心 [RetryPolicy](#) 类组成，而 `software.amazon.awssdk.core.retry` 包中 [software.amazon.awssdk.retries](#) 包含重试策略 API 元素。

重试策略 API 是作为统一核心组件接口和行为的 AWS 全局努力的一部分而引入的。SDKs

f SDK or Java 2.x 有三种内置的重试策略：标准、传统和自适应。所有三种重试策略都已预先配置为对一组可重试的异常进行重试。可重试错误的示例包括套接字超时、服务端限制、并发或乐观锁定失败以及暂时性服务错误。

标准重试策略

对于正常用例，建议RetryStrategy采用[标准重试策略](#)。与不同AdaptiveRetryStrategy，标准策略通常适用于所有重试用例。

默认情况下，标准重试策略执行以下操作。

- 根据构建时配置的条件进行重试。你可以用进行调整[StandardRetryStrategy.Builder#retryOnException](#)。
- 重试 2 次，总共尝试 3 次。你可以用进行调整[StandardRetryStrategy.Builder#maxAttempts\(int\)](#)。
- 使用[BackoffStrategy#exponentialDelay](#)退避策略，基本延迟为 100 毫秒，最大延迟为 20 秒。你可以用进行调整[StandardRetryStrategy.Builder#backoffStrategy](#)。
- 在下游故障频繁的情况下，执行断路（禁用重试）。总是执行第一次尝试，只禁用重试。调整为[StandardRetryStrategy.Builder#circuitBreakerEnabled](#)。

传统重试策略

[传统的重试策略](#)RetryStrategy适用于普通用例，但是，它已被弃用，转而使用StandardRetryStrategy当您不指定其他策略时，这是客户端使用的默认重试策略。

它的特点是区别对待限制和非限流异常，对于限制异常，退避的基本延迟（500 毫秒）大于非限制异常的基本延迟（100 毫秒），并且限制异常不会影响令牌桶状态。

在内部大规模使用这种策略的经验 AWS 表明，这并不比标准的重试策略好。此外，它无法保护下游服务免受重试风暴的影响，并可能导致客户端资源短缺。

默认情况下，旧版重试策略执行以下操作。

- 根据构建时配置的条件进行重试。你可以用进行调整[LegacyRetryStrategy.Builder#retryOnException](#)。
- 重试 3 次，总共尝试 4 次。你可以用进行调整[LegacyRetryStrategy.Builder#maxAttempts\(int\)](#)。

- 对于非限流异常，它使用`BackoffStrategy#exponentialDelay`退避策略，基本延迟为 100 毫秒，最大延迟为 20 秒。你可以通过以下方式进行调整：`RetryStrategy.Builder#backoffStrategy`。
- 对于限制异常，它使用`BackoffStrategy#exponentialDelay`退避策略，基本延迟为 500 毫秒，最大延迟为 20 秒。你可以用进行调整：`LegacyRetryStrategy.Builder#throttlingBackoffStrategy`。
- 在下游故障频繁的情况下，执行断路（禁用重试）。断路永远不会阻止第一次成功尝试。你可以使用来调整这种行为：`LegacyRetryStrategy.Builder#circuitBreakerEnabled`。
- 断路器的状态不受限流异常的影响。

自适应重试策略

自[自适应重试策略](#)`RetryStrategy`适用于资源限制程度高的用例。

自适应重试策略包括标准策略的所有功能，并添加了客户端速率限制器，用于衡量受限制请求与非限制请求的比率。该策略使用此衡量标准来减慢请求速度，以保持在安全的带宽范围内，理想情况下会导致零限制错误。

默认情况下，自适应重试策略执行以下操作。

- 根据构建时配置的条件进行重试。你可以用进行调整：`AdaptiveRetryStrategy.Builder#retryOnException`。
- 重试 2 次，总共尝试 3 次。你可以用进行调整：`AdaptiveRetryStrategy.Builder#maxAttempts(int)`。
- 使用动态退避延迟，该延迟基于下游资源的当前负载。
- 下游故障次数较多时，执行断路（禁用重试）。断路可能会阻止在中断情况下再次尝试以保护下游服务。

Warning

自适应重试策略假设客户端使用单个资源（例如，一个 DynamoDB 表或一个 Amazon S3 存储桶）。

如果您使用单个客户端管理多个资源，则在客户端访问所有其他资源时，与一个资源相关的限制或中断会导致延迟增加和故障。当您使用自适应重试策略时，我们建议您为每种资源使用一个客户端。

我们还建议您在所有客户端对资源使用自适应重试策略的情况下使用此策略。

⚠ Important

在 Java 2.26.0 版本中发布重试策略 SDK 包含了新的 `RetryMode.ADAPTIVE_V2` 枚举值。该 `ADAPTIVE_V2` 模式可以更正先前检测到限制错误时未能延迟第一次尝试的错误。

在 2.26.0 版本中，用户可以通过将 `ADAPTIVE_V2` 模式设置为 `adaptive` 环境变量、系统属性或配置文件设置来自动获得模式行为。这些设置没有 `adaptive_v2` 值。有关如何设置模式的信息，请参阅以下 [the section called “指定策略”](#) 部分。

用户可以通过使用在代码中设置模式来获得之前的行为 `RetryMode.ADAPTIVE`。

摘要：重试策略默认值的比较

下表显示了每种重试策略属性的默认值。

策略	最大尝试次数	非限流错误的基本延迟	限制错误的基本延迟	令牌桶大小	每次非限制重试的代币成本	每次限制重试的代币成本
Standard	3	100	100	500	5	5
传统	4	100	500	500	5	0
自适应	3	100	100	500	5	5

指定策略

您可以通过四种方式为服务客户端指定策略。

在代码中

在构建客户端时，您可以使用重试策略配置 `lambda` 表达式。以下代码段配置了在 `DynamoDB` 服务客户端上使用默认值的标准重试策略。

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

```
.build();
```

您可以指定`RetryMode.LEGACY`或`RetryMode.ADAPTIVE`代替`RetryMode.STANDARD`。

作为配置文件设置

`retry_mode`作为配置文件设置包含在[共享 AWS 配置文件中](#)。指定`standardlegacy`、或`adaptive`作为值。如果设置为配置文件设置，则在配置文件处于活动状态时创建的所有服务客户端都使用具有默认值的指定重试策略。您可以通过在代码中配置重试策略来覆盖此设置，如前所示。

使用以下配置文件，所有服务客户端都使用标准的重试策略。

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

作为JVM系统属性

您可以使用 `system` 属性为所有服务客户端配置重试策略，除非在代码中被覆盖。`aws.retryMode`指定`standardlegacy`、或`adaptive`作为值。

调用 Java 时使用 `-D` 开关，如以下命令所示。

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

或者，在创建任何客户端之前，在代码中设置系统属性，如以下代码段所示。

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

使用环境变量

也可以使用值为`standardlegacy`、或的`AWS_RETRY_MODE`环境变量`adaptive`。与配置文件设置或JVM系统属性一样，除非您在代码中配置客户端，否则环境变量会将所有服务客户端配置为指定的重试模式。

以下命令将当前 shell 会话`standard`的重试模式设置为。

```
export AWS_RETRY_MODE=standard
```

自定义策略

您可以通过设置最大尝试次数、退避策略和可重试的异常来自定义任何重试策略。您可以自定义何时构建重试策略或何时构建客户端，方法是使用允许进一步完善配置策略的覆盖生成器。

自定义最大尝试次数

您可以配置客户端构造期间的最大尝试次数，如以下语句所示。以下语句将客户端的默认重试策略自定义为最多 5 次尝试——第一次尝试加上 4 次重试。

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

或者，您可以构建策略并将其提供给客户端，如以下代码示例所示。以下代码将标准的最大 3 次尝试次数替换为 10 次，并使用自定义策略配置 DynamoDB 客户端。

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

Warning

我们建议您为每台客户机配置一个唯一的 `RetryStrategy` 实例。如果共享一个 `RetryStrategy` 实例，则一个客户端的失败可能会影响另一个客户端的重试行为。

您也可以使用[外部设置而不是代码来设置](#)所有客户端的最大尝试次数。您可以按照[the section called “指定策略”](#)节中所述配置此设置。

自定义可重试的异常

您可以配置其他异常，以便在客户端构建期间触发停用。此自定义是为引发的异常未包含在默认可重试异常集中的异常的边缘情况提供的。

以下代码片段显示了用于自定义重试异常的方法——`retryOnException`和。`retryOnExceptionOrCause`如果SDK引发直接异常或者异常被封装，则这些`retryOnExceptionOrCause`方法会添加一个可重试的异常。

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

自定义退避策略

您可以制定退避策略并将其提供给客户。

以下代码构建了一个`BackoffStrategy`替换默认标准策略的指数延迟退避策略的。

```
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();
```

从 `RetryPolicy` 迁移到 `RetryStrategy`

`RetryPolicy` (重试政策API) 将在可预见的将来得到支持。如果您当前使用的实例`RetryPolicy`来配置客户端，则一切都会像以前一样正常运行。在幕后，Java将其SDK改编成`RetryStrategy`了。新的重试策略接口提供的功能与 `a` 相同，`RetryPolicy`但创建和配置方式有所不同。

使用适用于 Java 的 SDK 2.x 进行日志记录

AWS SDK for Java 2.x 使用 [SLF4J](#)，这是一个抽象层，允许在运行时使用多个日志系统中的任何一个。

支持的日志记录系统包括 Java Logging Framework、Apache [Log4j 2](#) 和其他系统。本主题向您展示如何将 Log4j 2 作为与 SDK 结合使用的日志记录系统。

Log4j 2 配置文件

您通常是将一个名为 `log4j2.xml` 的配置文件与 Log4j 2 结合使用。配置文件示例如下所示。要了解有关配置文件中使用的值的更多信息，请参阅 [Log4j 配置手册](#)。

应用程序启动时，该 `log4j2.xml` 文件必须位于类路径中。对于 Maven 项目，请将文件放在 `<project-dir>/src/main/resources` 目录中。

`log4j2.xml` 配置文件会指定 [日志记录级别](#)、将日志记录输出发送到的位置（例如 [发送到文件或控制台](#)）和 [输出格式](#) 等属性。日志级别指定 Log4j 2 输出的详细级别。Log4j 2 支持多个日志记录 [层次结构](#) 的概念。可以为每级层次结构单独设置日志记录级别。与一起使用的主日志层次结构 AWS SDK for Java 2.x 是 `software.amazon.awssdk`。

添加日志记录依赖项

要在构建文件中配置 SLF4J 的 Log4j 2 绑定，请使用以下命令。

Maven

在 `pom.xml` 文件中添加以下元素：

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

Gradle–Kotlin DSL

在 `build.gradle.kts` 文件中添加以下内容。

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

对于 `log4j-slf4j2-impl` 构件的最低版本，请使用 `2.20.0`。要获取最新版本，请使用发布到 [Maven central](#) 的版本。将 `VERSION` 替换为您将要使用的版本。

特定于 SDK 的错误消息和警告

建议始终将“`software.amazon.awssdk`”记录器层次结构设置为“`WARN`”，以保证不会错过来自 SDK 客户端库的任何重要消息。例如，如果 Amazon S3 客户端检测到应用程序没有正确关闭 `InputStream` 而且可能会泄漏资源，那么 S3 客户端将通过向日志中记录警告消息来进行报告。另外，由此可确保客户端在处理请求或响应遇到任何问题时都会记录相应消息。

以下 `log4j2.xml` 文件将 `rootLogger` 设置为“`WARN`”，这会导致输出来自应用程序中所有记录器的警告和错误级别消息，包括“`software.amazon.awssdk`”层次结构中的记录器。如果使用 `<Root level="ERROR">`，也可将“`software.amazon.awssdk`”记录器层次结构显式设置为 `WARN`。

Log4j2.xml 配置文件示例

此配置会将所有记录器层次结构的“`ERROR`”和“`WARN`”级别的消息记录到控制台。

```
<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>
```

请求/响应摘要日志记录

对的每个请求都会 AWS 服务 生成一个唯一的 AWS 请求 ID，如果您在如何处理请求时遇到问题，AWS 服务 这会很有用。AWS 对于任何失败的服务调用，都可以通过软件开发工具包中的 [SdkServiceException](#) 对象以编程方式访问请求 ID，也可以通过“`software.amazon.awssdk.request`”记录器的“`调试`”日志级别报告请求 ID。

以下 `log4j2.xml` 文件将启用请求和响应的摘要。

```
<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>

```

以下是日志输出的示例：

```

2022-09-23 16:02:08 [main] DEBUG software.amazon.awssdk.request:85 - Sending Request:
DefaultSdkHttpRequest(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:
QS9DUMME2NHEDH8TGT9N5V530JV4KQNS05AEMVJF66Q9ASUAAJG, Extended Request ID: not
available

```

如果您只对请求编号感兴趣，请使用 `<Logger name="software.amazon.awssdk.requestId" level="DEBUG" />`。

调试级 SDK 日志

如果您需要有关 SDK 正在做什么的更多详细信息，可以将记录器的日志级别设置为 `software.amazon.awssdk DEBUG` 在此级别上，SDK 会输出大量细节，因此我们建议您设置此级别以使用集成测试来解决错误。

在此日志级别，SDK 会记录有关配置、凭据解析、执行拦截器、高级 TLS 活动、请求签名等的信息。

以下是 SDK 在 `S3Client#listBuckets()` 调用 `DEBUG` 级别上输出的语句示例。

```

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
...

```

以下log4j2.xml文件配置了之前的输出。

```

<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>

```

启用线路记录

查看适用于 Java 的 SDK 2.x 发送和接收的确切请求和响应可能很有用。如果您需要访问这些信息，可以根据服务客户端使用的 HTTP 客户端，通过添加必要的配置来启用它。

默认情况下，同步服务客户端（例如 [S3Client](#)）使用底层 [Apache HttpClient](#)，而异步服务客户端（例如 [S3](#)）使用 [Netty 非阻塞 AsyncClient](#) HTTP 客户端。

以下是可用于这两类服务客户端的 HTTP 客户端的细分：

同步 HTTP 客户端	异步 HTTP 客户端
ApacheHttpClient (默认)	NettyNioAsyncHttpClient (默认)
URLConnectionHttpClient	AwsCrtAsyncHttpClient

请参阅下面的相应标签，了解需要根据底层 HTTP 客户端添加的配置设置。

Warning

我们建议只出于调试目的使用线路日志记录。由于线路日志记录可能记录敏感数据，因此应在您的生产环境中禁用它。它会记录完整的请求或响应而不加密，即使对于 HTTPS 调用也是如此。对于大型请求（例如，将文件上传到 Amazon S3）或响应，详细的线路记录也会显著影响应用程序的性能。

ApacheHttpClient

将“org.apache.http.wire”记录器添加到 `log4j2.xml` 配置文件中，并将级别设置为“DEBUG”。

以下 `log4j2.xml` 文件开启了 Apache HttpClient 的完整线路记录。

```
<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>

```

使用 Apache 进行线路日志记录需要对 `log4j-1.2-api` 构件的额外 Maven 依赖项，因为它在后台使用 1.2。

以下构建文件片段显示了 `log4j 2` 的全套 Maven 依赖项，包括 Apache HTTP 客户端的线路日志记录。

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")
}
...
```

对于 `log4j-bom` 构件的最低版本，请使用 `2.20.0`。要获取最新版本，请使用发布到 [Maven central](#) 的版本。将 `VERSION` 替换为您将要使用的版本。

URLConnectionHttpClient

要记录使用 `URLConnectionHttpClient` 的服务客户端的详细信息，请先创建一个包含以下内容的 `logging.properties` 文件：

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

使用 `logging.properties` 的完整路径设置以下 JVM 系统属性：

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

此配置将仅记录请求和响应的标头，例如：

```
<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: sAFeZD0KdUMsBbkdjyDZw7P0oocb4C9KbiuzfJ6TWKQsGXHM/
```



```
dFu0vr2tUb7Y1wEHGdJ3DSIxq0={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}
```

要查看请求/响应正文，请将 `-Djavax.net.debug=all` 添加到 JVM 属性中。此附加属性记录了大量信息，包括所有 SSL 信息。

在日志控制台或日志文件中，搜索 "GET" 或 "POST" 以快速转到包含实际请求和响应的日志部分。使用 "Plaintext before ENCRYPTION" 搜索请求，使用 "Plaintext after DECRYPTION" 搜索响应，以查看标头和正文的全文。

NettyNioAsyncHttpClient

如果您的异步服务客户端使用默认值 `NettyNioAsyncHttpClient`，请在 `log4j2.xml` 文件中再添加两个记录器来记录 HTTP 标头和请求/响应正文。

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

以下是完整 `log4j2.xml` 示例：

```
<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>
```

这些设置记录所有标头详细信息和请求/响应正文。

AwsCrtAsyncHttpClient

如果您已将服务客户端配置为使用 `AwsCrtAsyncHttpClient` 的实例，则可以通过设置 JVM 系统属性或使用编程方式记录详细信息。

Log to a file at "Debug" level

使用系统属性：

```
-Daws.crt.log.level=Trace
-Daws.crt.log.destination=File
-Daws.crt.log.filename=<path to file>
```

使用编程方式：

```
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

使用系统属性：

```
-Daws.crt.log.level=Trace
-Daws.crt.log.destination=Stdout
```

使用编程方式：

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

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

出于安全考虑，在“跟踪”级别，`AwsCrtAsyncHttpClient` 仅记录响应标头，不记录请求标头、请求正文和响应正文。

为 DNS 名称查找设置 JVM TTL

Java 虚拟机 (JVM) 缓存 DNS 名称查找。当 JVM 将主机名解析为 IP 地址时，它会将该 IP 地址缓存一段指定的时间，即 time-to-live(TTL)。

由于 AWS 资源使用的 DNS 名称条目偶尔会发生变化，因此我们建议您将 JVM 的 TTL 值配置为 5 秒。这可确保在资源的 IP 地址发生更改时，您的应用程序将能够通过重新查询 DNS 来接收和使用资源的新 IP 地址。

对于一些 Java 配置，将设置 JVM 默认 TTL，以便在重新启动 JVM 之前绝不刷新 DNS 条目。因此，如果在应用程序仍在运行时 AWS 资源的 IP 地址发生变化，则在您手动重启 JVM 并刷新缓存的 IP 信息之前，它将无法使用该资源。在此情况下，设置 JVM 的 TTL，以便定期刷新其缓存的 IP 信息是极为重要的。

如何设置 JVM TTL

要修改 JVM 的 TTL，请设置 `net.workaddress.cache.ttl` 安全属性值，在 Java 8 的文件中设置该 `networkaddress.cache.ttl` 属性，在 Java 11 或更高版本 `$JAVA_HOME/jre/lib/security/java.security` 的文件中设置该属性。`$JAVA_HOME/conf/security/java.security`

以下是文件中的一段片段，该 `java.security` 文件显示 TTL 缓存设置为 5 秒。

```
#
# 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
...
```

在由 `$JAVA_HOME` 环境变量表示的 JVM 上运行的所有应用程序都使用此设置。

AWS SDK for Java 2.x 的最佳做法

本部分列出了使用适用于 Java 的 SDK 2.x 的最佳实践。

主题

- [如果可能，请重复使用 SDK 客户端](#)
- [关闭来自客户端操作的输入流](#)
- [根据性能测试调整 HTTP 配置](#)
- [对基于 Netty 的 HTTP 客户端使用 OpenSSL](#)

- [配置 API 超时](#)
- [使用指标](#)

如果可能，请重复使用 SDK 客户端

每个 SDK 客户端都维护自己的 HTTP 连接池。新请求可以重复使用池中已存在的连接，以缩短建立新连接的时间。我们建议共享单个客户端实例，以避免因未得到有效使用的连接池过多而产生的开销。所有 SDK 客户端都是线程安全的。

如果您不想共享客户端实例，请在不需要该客户端时在示例上调用 `close()` 以释放资源。

关闭来自客户端操作的输入流

对于流式传输操作（例如 [S3Client#getObject](#)），如果您直接使用 [ResponseInputStream](#)，建议执行以下操作：

- 尽快读取输入流中的所有数据。
- 请尽快关闭输入流。

我们之所以提出这些建议，是因为输入流是来自 HTTP 连接的直接数据流，并且在读取流中的所有数据并关闭流之前，底层 HTTP 连接无法重复使用。如果不遵守这些规则，则客户端可能会因为分配太多已打开但未使用的 HTTP 连接而耗尽资源。

根据性能测试调整 HTTP 配置

SDK 提供了一组适用于一般用例的[默认 http 配置](#)。我们建议客户根据其用例调整其应用程序的 HTTP 配置。

作为一个很好的起点，SDK 提供了[智能配置默认值](#)功能。此功能从版本 2.17.102 开始提供。根据您的用例选择一种模式，该模式将提供合理的配置值。

对基于 Netty 的 HTTP 客户端使用 OpenSSL

默认情况下，SDK 的 [NettyNioAsyncHttpClient](#) 使用 JDK 的默认 SSL 实现作为 `SslProvider`。我们的测试发现，OpenSSL 的性能比 JDK 的默认实现要好。Netty 社区也[建议使用 OpenSSL](#)。

要使用 OpenSSL，请将 `netty-tcnative` 添加到您的依赖项中。有关配置的详细信息，请参阅 [Netty 项目文档](#)。

在为项目配置了 `netty-tcnative` 后，`NettyNioAsyncHttpClient` 实例将自动选择 `OpenSSL`。或者，您可以使用 `NettyNioAsyncHttpClient` 生成器显式设置 `SslProvider`，如以下代码段所示。

```
NettyNioAsyncHttpClient.builder()
    .sslProvider(SslProvider.OPENSSL)
    .build();
```

配置 API 超时

SDK 为某些超时选项（例如连接超时和套接字超时）提供**默认值**，但不为 API 调用超时或单个 API 调用尝试超时提供默认值。为单个尝试和整个请求都设置超时是一种很好的做法。当存在可能导致请求尝试花费更长时间才能完成的临时问题或出现严重的网络问题时，这将确保您的应用程序以最佳方式快速失败。

您可以使用 [ClientOverrideConfiguration#apiCallAttemptTimeout](#) 和 [ClientOverrideConfiguration#apiCallTimeout](#) 为服务客户端发出的所有请求配置超时。

以下示例演示使用自定义超时值配置 Amazon S3 客户端。

```
S3Client.builder()
    .overrideConfiguration(
        b -> b.apiCallTimeout(Duration.ofSeconds(<custom value>))
            .apiCallAttemptTimeout(Duration.ofMillis(<custom value>)))
    .build();
```

apiCallAttemptTimeout

此设置设定单次 HTTP 尝试的时长，超过该时长之后可以重试 API 调用。

apiCallTimeout

此属性的值配置整个执行的时间，包括所有重试尝试。

除了在服务客户端上设置这些超时值之外，您还可以使用 [RequestOverrideConfiguration#apiCallTimeout\(\)](#) 和 [RequestOverrideConfiguration#apiCallAttemptTimeout\(\)](#) 来配置单个请求。

以下示例使用自定义超时值配置单个 `listBuckets` 请求。

```
s3Client.listBuckets(lbr -> lbr.overrideConfiguration(
```

```
b -> b.apiCallTimeout(Duration.ofSeconds(<custom value>))
    .apiCallAttemptTimeout(Duration.ofMillis(<custom value>))));
```

将这些属性一起使用时，可以对所有重试尝试所花费的总时间设置硬性限制。您还可以将单个 HTTP 请求设置为在慢速请求时快速失败。

使用指标

适用于 Java 的 SDK 可以[收集应用程序中服务客户端的指标](#)。您可以使用这些指标来识别性能问题、查看总体使用趋势、查看返回的服务客户端异常或深入了解特定问题。

我们建议您收集指标，然后分析 Amazon CloudWatch Logs，以便更深入地了解应用程序的性能。

问题排查常见问题

在应用程序 AWS SDK for Java 2.x 中使用时，可能会遇到本主题中列出的运行时错误。使用此处的建议来帮助您找出根本原因并解决错误。

如何修复“`java.net.SocketException：连接重置`”或“服务器无法完成响应”错误？

连接重置错误表示您的主机 AWS 服务、或任何中间方（例如，NAT 网关、代理、负载均衡器）在请求完成之前关闭了连接。由于原因有很多，因此要找到解决方案，就必须知道连接关闭的原因。以下各项通常会导致连接关闭。

- 连接处于非活动状态。这在流媒体操作中很常见，在这种操作中，数据在一段时间内没有写入或写出电线，因此中间方检测到连接已失效并关闭了连接。为防止出现这种情况，请确保您的应用程序主动下载或上传数据。
- 你已经关闭了 HTTP 或 SDK 客户端。确保不要在资源使用期间将其关闭。
- 代理配置错误。尝试绕过您配置的任何代理，以查看它是否可以解决问题。如果这样可以解决问题，则代理出于某种原因正在关闭您的连接。研究您的特定代理以确定它关闭连接的原因。

如果您无法识别问题，请尝试在网络的客户端边缘为受影响的连接运行 TCP 转储（例如，在您控制的任何代理之后）。

如果您看到 AWS 端点正在发送 TCP RST（重置），[请联系受影响的服务](#)，看看他们能否确定重置的原因。请准备好提供请求编号和问题发生时间的戳。AWS 支持团队还可能受益于[电汇日志](#)，[这些日志](#)可以准确显示您的应用程序发送和接收的字节以及何时发送。

如何修复“连接超时”？

连接超时错误表示您的主机 AWS 服务、或任何中间方（例如，NAT 网关、代理、负载均衡器）未能在配置的连接超时内与服务器建立新连接。以下内容描述了此问题的常见原因。

- 配置的连接超时过低。默认情况下，连接超时为 2 秒 AWS SDK for Java 2.x。如果将连接超时设置得太低，则可能会出现此错误。如果您只进行区域内呼叫，则建议的连接超时为 1 秒；如果您发出跨区域请求，则建议的连接超时时间为 3 秒。
- 代理配置错误。尝试绕过您配置的任何代理，以查看它是否可以解决问题。如果这样可以解决问题，则代理是连接超时的原因。研究您的特定代理以确定发生这种情况的原因

如果您无法识别问题，请尝试在网络的客户端边缘为受影响的连接运行 TCP 转储（例如，在您控制的任何代理之后），以调查任何网络问题。

如何修复“`java.net.SocketTimeoutException`：读取超时”？

读取超时错误表示 JVM 尝试从底层操作系统读取数据，但未在通过 SDK 配置的时间内返回数据。如果操作系统、或任何中间方（例如 AWS 服务，NAT 网关、代理、负载均衡器）未能在 JVM 预期的时间内发送数据，则可能会发生此错误。由于原因有很多，因此要找到解决方案，就必须知道未返回数据的原因。

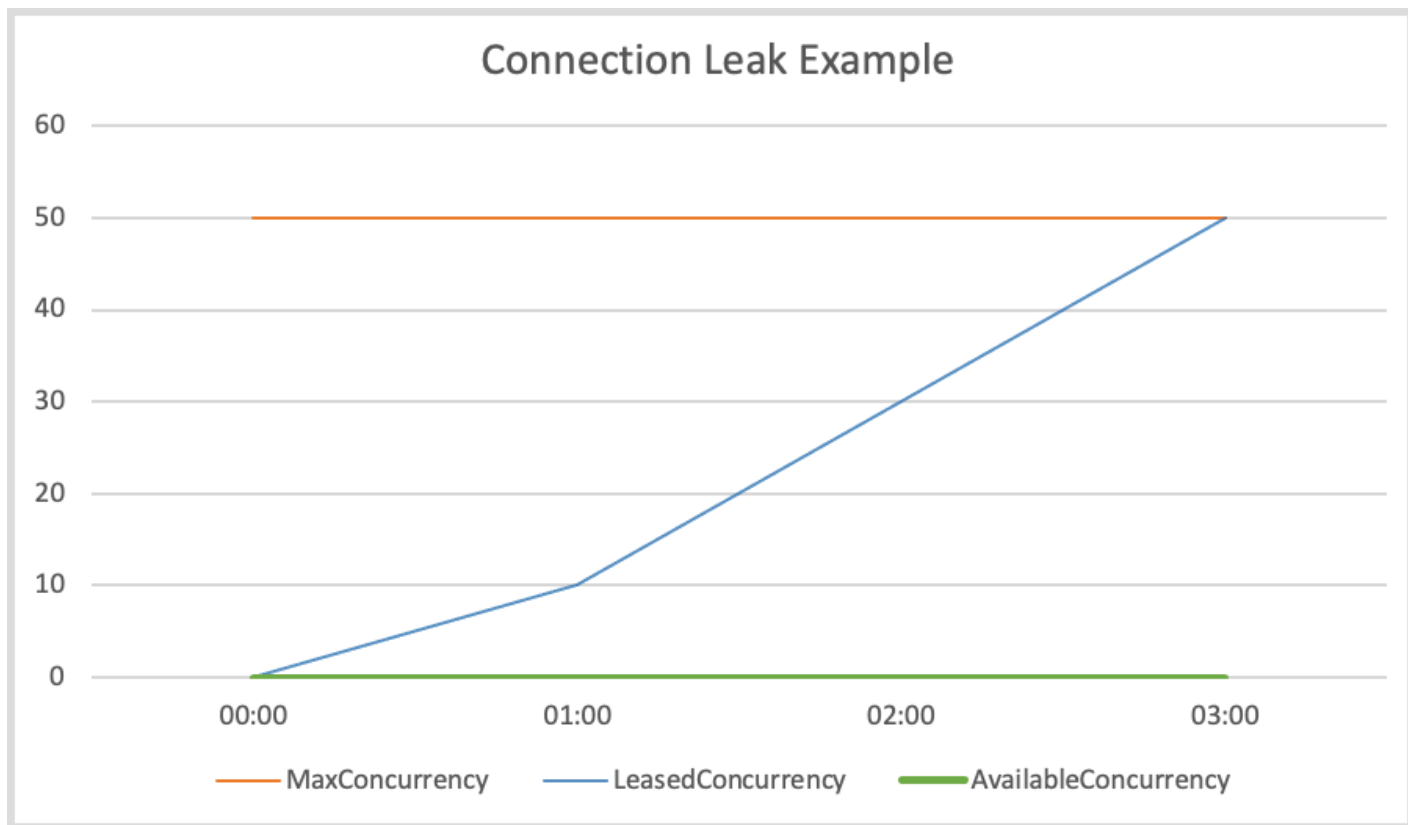
尝试在网络的客户端边缘为受影响的连接运行 TCP 转储（例如，在您控制的任何代理之后）。

如果您看到 AWS 终端节点正在发送 TCP RST（重置），[请联系受影响的服务](#)。请准备好提供请求编号和问题发生时间的戳。AWS 支持团队还可能受益于[电汇日志](#)，[这些日志](#)可以准确显示您的应用程序发送和接收的字节以及何时发送。

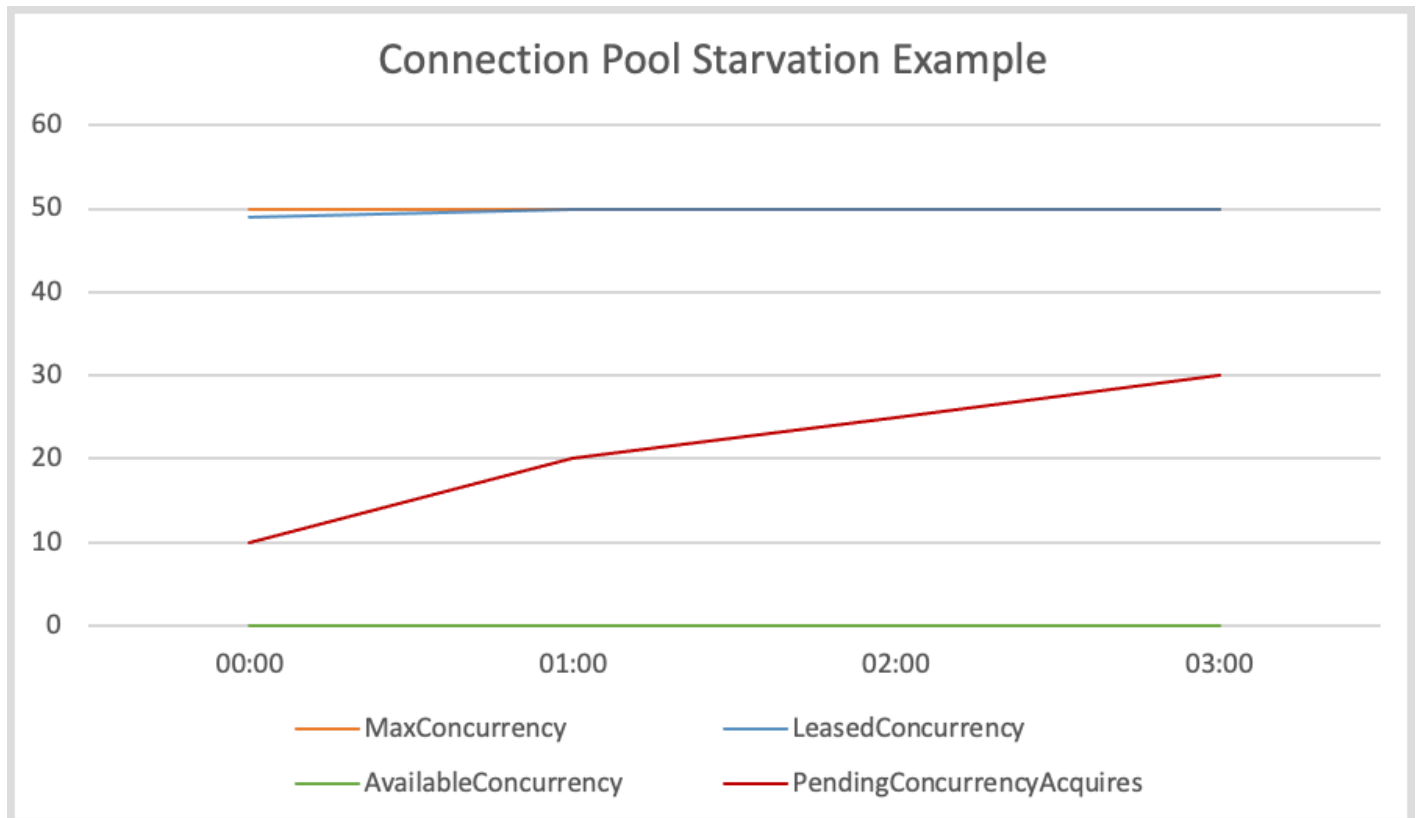
如何修复“无法执行 HTTP 请求：等待池连接超时”错误？

此错误表示请求无法在指定的最长时间从池中获取连接。要解决问题，我们建议您[启用 SDK 客户端指标](#)，以便向 Amazon CloudWatch 发布指标。HTTP 指标可以帮助缩小根本原因范围。以下项目描述了此错误的常见原因。

- 连接泄漏。您可以通过检查 `LeasedConcurrencyAvailableConcurrency`、`MaxConcurrency` 指标来对此进行调查。如果在到达之前一直 `LeasedConcurrency` 增加 `MaxConcurrency` 但从未降低，则可能存在连接泄漏。泄漏的常见原因是流媒体操作（例如 `S3 getObject` 方法）未关闭。我们建议您的应用程序尽快从输入流中读取所有数据，[然后关闭输入流](#)。下图显示了连接泄漏的 SDK 指标可能是什么样子。



- 连接池不足。如果您的请求速率过高，并且已配置的连接池大小无法满足请求需求，则可能会发生这种情况。默认连接池大小为 50，当池中的连接达到最大值时，HTTP 客户端会将传入的请求排队，直到连接可用为止。下图显示了连接池不足的 SDK 指标可能是什么样子。



要缓解此问题，请考虑采取以下任何措施。

- 增加连接池的大小，
- 增加采集超时时间。
- 降低请求速率。

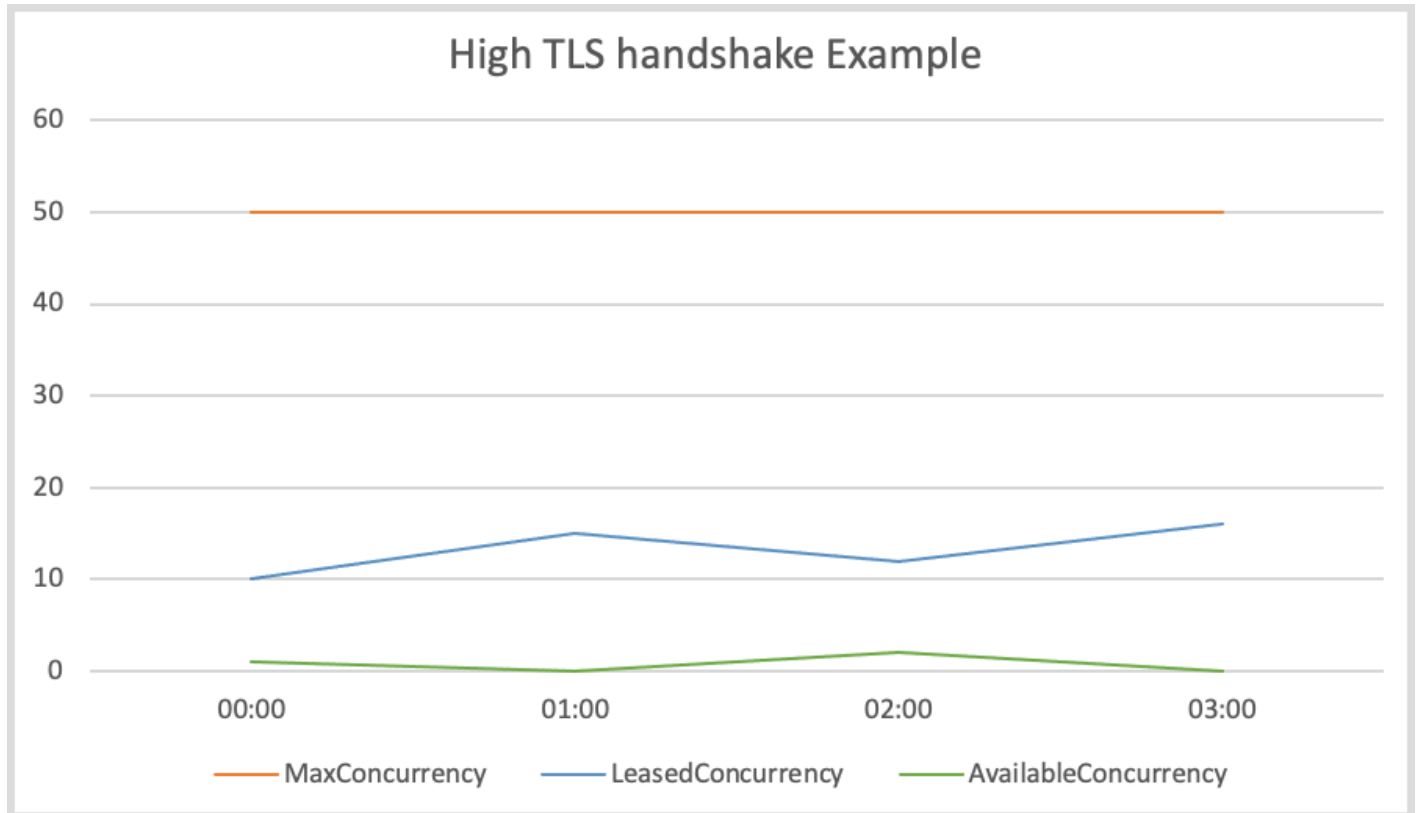
通过增加最大连接数，可以增加客户端吞吐量（除非网络接口已被充分利用）。但是，您最终可能会遇到操作系统对进程使用的文件描述符数量的限制。如果您已经完全使用网络接口或无法进一步增加连接数，请尝试延长获取超时时间。随着时间的增加，在超时之前，您可以获得更多时间来请求获取连接。如果连接没有释放，则后续请求仍将超时。

如果您无法通过使用前两种机制来解决问题，请尝试以下选项来降低请求速率。

- 平滑您的请求，以免大量流量爆发使客户端超负荷。
- 通过呼叫提高效率 AWS 服务。
- 增加发送请求的主机数量。
- I/O 线程太忙了。这仅适用于您使用带的异步 SDK 客户端 [NettyNioAsyncHttpClient](#)。如果该 AvailableConcurrency 指标不低（表示池中存在连接），而是很高，ConcurrencyAcquireDuration 则可能是因为 I/O 线程无法处理请求。确保你不

是 `Runnable:run` 作为 [未来完成执行器传递的](#)，也不要 [在响应的未来完成链中](#) 执行耗时的任务，因为这可能会阻塞 I/O 线程。如果不是这样，请考虑使用该 [eventLoopGroupBuilder](#) 方法增加 I/O 线程的数量。作为参考，`NettyNioAsyncHttpClient` 实例的默认 I/O 线程数是主机 CPU 核心数的两倍。

- TLS 握手延迟高。如果您的 `AvailableConcurrency` 指标接近 0 且 `LeasedConcurrency` 于 `MaxConcurrency`，则可能是因为 TLS 握手延迟很高。下图显示了高 TLS 握手延迟的 SDK 指标可能是什么样子。



对于 Java SDK 提供的不基于 CRT 的 HTTP 客户端，请尝试启用 [TLS 日志](#) 来解决 TLS 问题。对于 AWS 基于 CRT 的 HTTP 客户端，请尝试启用 [AWS CRT](#) 日志。如果您发现 AWS 端点似乎需要很长时间才能执行 TLS 握手，则应 [联系受影响的服务](#)。

我该如何修

复 `NoClassDefFoundError`，`NoSuchMethodError` 或 `NoSuchFieldError`？

A `NoClassDefFoundError` 表示无法在运行时加载类。导致此错误的两个最常见原因是：

- 该类在类路径中不存在，因为 JAR 丢失或类路径上的 JAR 版本不正确。
- 该类无法加载，因为其静态初始化器引发了异常。

同样，`NoSuchMethodErrors` 和 `NoSuchFieldErrors` 通常是由于 JAR 版本不匹配所致。我们建议您执行以下步骤。

1. 检查你的依赖关系，确保你使用的所有 SDK jar 版本相同。找不到类、方法或字段的最常见原因是升级到新的客户端版本，但继续使用旧的“共享”SDK 依赖版本。新的客户端版本可能会尝试使用仅存在于较新的“共享”SDK 依赖项中的类。尝试运行 `mvn dependency:tree` 或 `gradle dependencies`（对于 Gradle）来验证 SDK 库版本是否全部匹配。为了完全避免将来出现此问题，我们建议使用 [BOM（物料清单）](#) 来管理 SDK 模块版本。

以下示例向您展示了混合 SDK 版本的示例。

```
[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
```

的版本 `dynamodb` 是 2.20.00，的版本 `aws-core` 是 2.13.19。`aws-core` 工件版本也应为 2.20.00。

2. 在日志的早期检查语句，以查看某个类是否由于静态初始化失败而无法加载。当类第一次加载失败时，它可能会抛出一个不同的、更有用的异常来指定无法加载该类的原因。这个可能有用的异常只发生一次，因此以后的日志语句只会报告未找到该类。
3. 检查您的部署过程，确保它实际上与您的应用程序一起部署了所需的 JAR 文件。您可能使用正确的版本进行构建，但是为应用程序创建类路径的过程排除了必需的依赖项。

如何修复“`SignatureDoesNotMatch`”错误或“我们计算的请求签名与您提供的签名不匹配”错误？

`SignatureDoesNotMatch` 错误表示生成的签名 AWS SDK for Java 和生成的签名 AWS 服务 不匹配。以下项目描述了潜在原因。

- 代理方或中间方修改请求。例如，代理或负载均衡器可能会修改由 SDK 签名的标头、路径或查询字符串。
- 服务和 SDK 的不同之处在于它们在生成待签字符串时对请求进行编码的方式。

要调试此问题，我们建议您为 SDK [启用调试日志记录](#)。尝试重现错误并找到 SDK 生成的规范请求。在日志中，规范请求用 `AWS4 Canonical Request: ...` 标记，待签字符串标记 `AWS4 String to sign: ...`

如果您无法启用调试（例如，因为它只能在生产环境中重现），请向应用程序添加逻辑，以便在错误发生时记录有关请求的信息。然后，您可以使用该信息尝试在启用调试日志记录的集成测试中将错误复制到生产环境之外。

收集规范请求和待签字符串后，将其与 [Signature 版本 4 规范进行比较](#)，以确定 SDK 生成待 AWS 签字符串的方式是否存在问题。如果出现问题，可以创建 [GitHub 错误报告](#) 给 AWS SDK for Java。

如果没有出现任何问题，您可以将 SDK 的待签字符串与失败响应中 AWS 服务返回的待签字符串（例如 Amazon S3）进行比较。如果这不可用，则应 [联系受影响的服务](#)，以查看它们生成了哪些规范请求和待签字符串，以便进行比较。这些比较可以帮助识别可能修改了请求或服务与客户端之间的编码差异的中间方。

有关签署请求的更多背景信息，请参阅 AWS Identity and Access Management 用户指南中的 [签署 AWS API 请求](#)。

Example 规范请求的

```
PUT
/Example-Bucket/Example-Object
partNumber=19&uploadId=string
amz-sdk-invocation-id:f8c2799d-367c-f024-e8fa-6ad6d0a1afb9
amz-sdk-request:attempt=1; max=4
content-encoding:aws-chunked
content-length:51
content-type:application/octet-stream
host:xxxxx
x-amz-content-sha256:STREAMING-UNSIGNED-PAYLOAD-TRAILER
x-amz-date:20240308T034733Z
x-amz-decoded-content-length:10
x-amz-sdk-checksum-algorithm:CRC32
x-amz-trailer:x-amz-checksum-crc32
```

Example 待签字符串

```
AWS4-HMAC-SHA256
20240308T034435Z
20240308/us-east-1/s3/aws4_request
5f20a7604b1ef65dd89c333fd66736fdef9578d11a4f5d22d289597c387dc713
```

如何修复“`java.lang.IllegalStateException`：连接池关闭”错误？

此错误表示底层 Apache HTTP 连接池已关闭。以下项目描述了潜在原因。

- SDK 客户端过早关闭。只有在关联的客户端关闭时，SDK 才会关闭连接池。确保不要在资源使用期间将其关闭。
- A **java.lang.Error** 被扔了。诸如此类的错误 `OutOfMemoryError` 会导致 Apache HTTP 连接池 [关闭](#)。检查您的日志中是否有错误堆栈痕迹。还要检查你的代码，看看它捕获 `Throwable`s 或 `Errors` 但会吞下输出以防止错误浮出水面的地方。如果您的代码未报告错误，请重写代码以记录信息。记录的信息有助于确定错误的根本原因。
- 您尝试使用在证书提供程序关闭 `DefaultCredentialsProvider#create()` 后返回的凭证提供程序。 [DefaultCredentialsProvider#create](#) 返回一个单例实例，因此，如果它已关闭并且您的代码调用了该 `resolveCredentials` 方法，则会在缓存的凭证（或令牌）过期后引发异常。

检查您的代码中 `DefaultCredentialsProvider` 是否有关闭的地方，如以下示例所示。

- 通过调用关闭单例实例 `DefaultCredentialsProvider#close()`。

```
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();
```

- 在 `try-with-resources` 方块 `DefaultCredentialsProvider#create()` 中调用。

```
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();
```

`DefaultCredentialsProvider.builder().build()`如果您的代码已关闭单例实例，并且您需要使用解析证书，则通过调用来创建一个新的非单例实例。`DefaultCredentialsProvider`

使用 AWS SDK for Java 2.x 的功能

一般功能

SDK适用于 Java 2.x 的 2.x 包含多项功能，这些功能使编程 AWS 服务 变得更加容易。

- SDK隐藏了[检索分页结果](#)和[轮询](#)资源背后的复杂机制。
- [使用非阻塞 I/O 进行异步编程](#)可帮助您编写性能更高的并发代码。SDK提供 [HTTP/2](#) 的好处，例如尽可能减少延迟。
- Java SDK 可以生成[指标](#)来帮助您监控应用程序的运行状况。

特定于服务的功能

除了前面提到的一般功能外，Java SDK 还提供了特定功能 AWS 服务。

- Amazon S3 — 为了[简化您使用 Amazon S3 处理文件和目录的工作](#)，SDK提供了 S3 传输管理器。为了在使用标准异步 S3 时[提高性能和可靠性](#)API，SDK提供了 AWS CRT 基于的 S3 客户端。SDK
- D ynamoDB — [面向对象的映射](#)功能由 DynamoDB 增强版客户端提供。API使用“增强文档”[处理以文档为导向的JSON样式数据](#)。API
- IAM— IAM 策略生成器API提供了一种[类型安全、面向对象的方式来](#)创建策略。IAM

使用 AWS SDK for Java 2.x 处理分页结果

当响应对象太大而无法在单个响应中返回时，许多 AWS 操作都会返回分页结果。在 AWS SDK for Java 1.0 中，响应包含一个用于检索下一页结果的标记。相比之下，AWS SDK for Java 2.x 具有自动分页方法，可以进行多次服务调用，自动为您获取下一页的结果。您只需编写处理结果的代码。自动分页功能适用于同步和异步客户端。

Note

这些代码片段假设您了解[使用的基础知识SDK](#)，并且已将您的环境配置为[单点登录访问权限](#)。

同步分页

以下示例演示列出 Amazon S3 桶中对象的同步分页方法。

迭代页面

第一个示例演示了如何使用分页器对象（一个 [ListObjectsV2Iterable](#) 实例）来使用该方法遍历所有响应页面。stream 代码在响应页面上进行流式传输，将响应流转换为 [S3Object](#) 内容流，然后处理 Amazon S3 对象的内容。

以下导入适用于此同步分页部分中的所有示例。

导入

```
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()));

```

请参阅上的[完整示例](#) GitHub。

迭代对象

以下示例演示了迭代响应中返回的对象（而不是响应的页面）的方法。ListObjectsV2Iterable 类的 contents 方法返回一个 [SdkIterable](#)，它提供了几种处理底层内容元素的方法。

使用流

以下代码段在响应内容上使用 stream 方法来迭代分页项目集合。

```

    // Helper method to work with paginated collection of items directly.
    listRes.contents().stream()
        .forEach(content -> System.out
            .println(" Key: " + content.key() + "
size = " + content.size()));

```

请参阅上的[完整示例](#) GitHub。

使用 for-each 循环

由于 SdkIterable 扩展了 Iterable 接口，因此您可以像处理任何 Iterable 一样处理内容。以下代码段使用标准 for-each 循环迭代响应的内容。

```

    for (S3Object content : listRes.contents()) {
        System.out.println(" Key: " + content.key() + " size = " +
content.size());
    }

```

请参阅上的[完整示例](#) GitHub。

手动分页

如果您的使用案例需要手动分页，则手动分页仍然可用。对后续请求使用响应对象中的下一个令牌。以下示例使用 while 循环。

```
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();
}
```

请参阅上的[完整示例](#) GitHub。

异步分页

以下示例演示了列出 DynamoDB 表格的异步分页方法。

迭代表名称页面

以下两个示例使用异步 DynamoDB 客户端，该客户端调用listTablesPaginator该方法并请求获取。[ListTablesPublisher](#) ListTablesPublisher实现了两个接口，这为处理响应提供了许多选项。我们将研究每个接口的方法。

使用 **Subscriber**

以下代码示例演示如何使用 `ListTablesPublisher` 实现的 `org.reactivestreams.Publisher` 接口处理分页结果。要了解有关响应式流模型的更多信息，请参阅 [React ive St GitHub reams 存储库](#)。

以下导入适用于此异步分页部分中的所有示例。

导入

```
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;
```

以下代码获取一个 `ListTablesPublisher` 实例。

```
// 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);
```

以下代码使用 `org.reactivestreams.Subscriber` 的匿名实现来处理每个页面的结果。

`onSubscribe` 方法将调用 `Subscription.request` 方法来对来自发布者的数据启动请求。必须调用此方法以开始从发布者获取数据。

订阅者的 `onNext` 方法将处理响应页面，它会访问所有表名称并打印出每个表名称。处理完该页面后，会向发布者请求另一个页面。将重复调用该方法，直到检索了所有页面。

如果检索数据时出现错误，将触发 `onError` 方法。最后，在 `onComplete` 方法在请求所有页面后调用。

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

请参阅上的[完整示例](#) GitHub。

使用 **Consumer**

ListTablesPublisher 实现的 SdkPublisher 接口有一个 subscribe 方法，该方法接受 Consumer 并返回 CompletableFuture<Void>。

此接口中的 subscribe 方法可用于 org.reactivestreams.Subscriber 开销可能过大的简单用例。当下面的代码使用每个页面时，它会在每个页面上调用 tableNames 方法。该 tableNames 方法返回使用 forEach 方法处理的 DynamoDB 表名的 java.util.List。

```
// Use a Consumer for simple use cases.
CompletableFuture<Void> future = publisher.subscribe(
    response -> response.tableNames()
        .forEach(System.out::println));
```

请参阅上的[完整示例](#) GitHub。

迭代表名称

以下示例演示了迭代响应中返回的对象（而不是响应的页面）的方法。与之前用 contents 方法演示的同步 Amazon S3 示例类似，DynamoDB 异步结果类 ListTablesPublisher 具有与底层项目集合交互的 tableNames 便捷方法。此 tableNames 方法的返回类型是一个 [SdkPublisher](#)，可用于跨所有页面请求项目。

使用 **Subscriber**

以下代码获取表名底层集合的 SdkPublisher。

```
// 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();
```

以下代码使用 org.reactivestreams.Subscriber 的匿名实现来处理每个页面的结果。

订阅者的 `onNext` 方法将处理集合中的单个元素。在本例中，它是一个表名称。处理完该表名称后，会向发布者请求另一个表名称。将重复调用该方法，直到检索了所有表名称。

```
// 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() {
    }
});
```

请参阅上的[完整示例](#) GitHub。

使用 **Consumer**

以下示例使用 `SdkPublisher` 的 `subscribe` 方法（采用 `Consumer`）来处理每个项目。

```
// Use a Consumer.
CompletableFuture<Void> future = publisher.subscribe(System.out::println);
future.get();
```

请参阅上的[完整示例](#) GitHub。

使用第三方库

您可以使用其他第三方库，而不是实现自定义订阅者。此示例演示了用法 RxJava，但是可以使用任何实现响应式流接口的库。有关该库的更多信息，[GitHub 请参阅上的 RxJava wiki 页面](#)。

要使用该库，请将其作为依赖项添加。如果使用 Maven，则该示例将显示要使用的 POM 代码片段。

POM 参赛作品

```
<dependency>
  <groupId>io.reactivex.rxjava3</groupId>
  <artifactId>rxjava</artifactId>
  <version>3.1.6</version>
</dependency>
```

代码

```
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::tableName)
    .toList()
    .blockingGet();
System.out.println(tables);
```

请参阅上的[完整示例](#) GitHub。

在 AWS SDK for Java 2.x 中对资源状态进行民意调查：服务员

AWS SDK for Java 2.x 的 waiters 实用程序使您能够在对这些 AWS 资源执行操作之前验证这些资源是否处于指定状态。

服务员是一种抽象概念，用于轮询 AWS 资源，例如 DynamoDB 表或 Amazon S3 存储桶，直到达到所需的状态（或者直到确定资源永远无法达到所需状态）。与其编写逻辑来持续轮询 AWS 资源（这可能很繁琐且容易出错），不如使用服务员来轮询资源，让您的代码在资源准备就绪后继续运行。

先决条件

必须先完成[设置 AWS SDK for Java 2.x](#) 中的步骤 `AWS SDK for Java`，然后才能在项目中使用服务

您还必须将项目依赖项（例如，在您的 `pom.xml` 或 `build.gradle` 文件中）配置为使用 `AWS SDK for Java` 版本 `2.15.0` 或更高版本。

例如：

```
<project>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>2.15.0</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
</project>
```

使用 Waiter

要实例化 `waiter` 对象，请先创建一个服务客户端。将服务客户端的 `waiter()` 方法设置为 `waiter` 对象的值。`waiter` 实例存在后，设置其响应选项以执行相应的代码。

同步编程

以下代码片段显示了如何等待 `DynamoDB` 表存在并处于 `ACTIVE` 状态。

```
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);
```


异步编程

以下代码片段显示了如何等待 DynamoDB 表不再存在。

```
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();
```

配置 Waiter

您可以使用 waiter 的生成器上的 `overrideConfiguration()`，为 waiter 自定义配置。对于某些操作，您可以在发出请求时应用自定义配置。

配置 Waiter

以下代码段演示如何覆盖 waiter 的配置。

```
// 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();
```

覆盖特定请求的配置

以下代码段演示如何根据每个请求覆盖 waiter 的配置。请注意，只有某些操作具有可自定义的配置。

```
waiter.waitForTableNotExists(b -> b.tableName("myTable"),
    o -> o.maxAttempts(10));

asyncWaiter.waitForTableExists(b -> b.tableName("myTable"),
    o -> o.waitFor(Duration.ofMinutes(1)));
```

代码示例

有关将 waiters 与一起使用的完整示例 DynamoDB，请参阅 AWS 代码示例[CreateTable存储库中的.java](#)。

有关使用服务员的完整示例 Amazon S3，请参阅 AWS 代码示例[存储库中的 S3 BucketOps.java](#)。

使用异步编程

这些 AWS SDK for Java 2.x 特点是支持非阻塞 I/O 的异步客户端，可在几个线程之间实现高并发性。但是，不能保证总的非阻塞 I/O。在某些情况下，异步客户端可能会执行阻塞调用，例如凭据检索、使用[AWS 签名版本 4 \(Sigv4\)](#) 进行请求签名或端点发现。

同步方法会阻止执行您的线程，直到客户端接收到服务的响应。异步方法会立即返回，并控制调用的线程，而不必等待响应。

由于异步方法在收到响应之前返回，所以需要某种方法在响应准备就绪时接收响应。2.x 中 AWS SDK for Java 返回 `CompletableFuture` 对象中的异步客户端方法，允许您在响应准备就绪时访问响应。

非流式操作

对于非流式操作，异步方法调用类似于同步方法。但是，中的异步方法会 AWS SDK for Java 返回一个包含将来异步操作结果的 `CompletableFuture` 对象。

当结果可用时，使用要完成的操作调用 `CompletableFuture whenComplete()` 方法。`CompletableFuture` 实现 `Future` 接口，以便您还可以通过调用 `get()` 方法来获得响应对象。

以下是一个异步操作的示例，该操作调用一个 Amazon DynamoDB 函数来获取表列表，接收 `CompletableFuture` 可以容纳 [ListTablesResponse](#) 对象的。在调用 `whenComplete()` 时定义的操作仅在异步调用完成时完成。

导入

```
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;
```

代码

```
public class DynamoDBAsyncListTables {

    public static void main(String[] args) throws InterruptedException {

        // Create the DynamoDbAsyncClient object
        Region region = Region.US_EAST_1;
        DynamoDbAsyncClient client = DynamoDbAsyncClient.builder()
            .region(region)
            .build();

        listTables(client);
    }

    public static void listTables(DynamoDbAsyncClient client) {

        CompletableFuture<ListTablesResponse> response =
client.listTables(ListTablesRequest.builder()
            .build());

        // Map the response to another CompletableFuture containing just the table
names
        CompletableFuture<List<String>> tableNames =
response.thenApply(ListTablesResponse::tableNames);

        // When future is complete (either successfully or in error) handle the
response
        tableNames.whenComplete((tables, err) -> {
            try {
                if (tables != null) {
                    tables.forEach(System.out::println);
                } else {
                    // Handle error
                }
            }
        });
    }
}
```

```

        err.printStackTrace();
    }
    } finally {
        // Lets the application shut down. Only close the client when you are
        completely done with it.
        client.close();
    }
    });
    tableNames.join();
}
}
}

```

以下代码示例说明如何使用异步客户端从表中检索项目。调用 `getItem` 的方法，`DynamoDbAsyncClient` 然后向其传递一个包含所需项目的表名和主键值的 [GetItemRequest](#) 对象。这通常是您传递此操作所需数据的方式。在此示例中，请注意传递了一个字符串值。

导入

```

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;

```

代码

```

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);
}
```

请参阅上的[完整示例](#) GitHub。

流式操作

对于流媒体操作，您必须提供[AsyncRequestBody](#)以增量方式提供内容，或者提供[AsyncResponseTransformer](#)以接收和处理响应。

以下示例使用操作将文件 Amazon S3 异步上传到。PutObject

导入

```
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;
```

代码

```
/**
```

```
* 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();
}
}
```

以下示例使用GetObject操作从 Amazon S3 异步获取文件。

导入

```
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;
```

代码

```
/**
 * 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();
}
}
```

高级操作

AWS SDK for Java 2.x 使用 [Netty](#) (一种异步事件驱动的网络应用程序框架) 来处理 I/O 线程。AWS SDK for Java 2.x 创建了一个 `ExecutorService` 落后的 `Netty`，以完成从 HTTP 客户端请求返回到 `Netty` 客户端的期货。这种抽象化可以减少在客服人员选择停止或休眠线程时，应用程序中断异步处理的风险。默认情况下，每个异步客户端都会根据处理器数量创建一个线程池，并管理 `ExecutorService` 队列中的任务。

创建异步客户端时，高级用户可在构建时使用以下选项指定其线程池大小。

代码

```
S3AsyncClient clientThread = S3AsyncClient.builder()
    .asyncConfiguration(
        b -> b.advancedOption(SdkAdvancedAsyncClientOption
            .FUTURE_COMPLETION_EXECUTOR,
            Executors.newFixedThreadPool(10))
    )
)
.build();
```

要优化性能，您可以管理自己的线程池执行程序，并在配置客户端时包括它。

```
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();
```

在 HTTP /2 中使用 AWS SDK for Java

HTTP/2 是 HTTP 协议的主要修订版。这一新版本具有多个增强功能以提高性能：

- 二进制数据编码提供了更高效的数据传输。
- 标头压缩可减少客户端下载的开销字节数，同时帮助客户端更快地获取内容。这对于受带宽限制的移动客户端尤其有用。
- 双向异步通信（多路复用）允许客户端之间同时通过单个连接而不是通过多个连接传送多个请求和 AWS 响应消息，从而提高性能。

升级到最新版本的开发者 SDKs 将自动使用 HTTP /2，前提是他们正在使用的服务支持该版本。新的编程接口可以无缝利用 HTTP /2 功能，并提供构建应用程序的新方法。

AWS SDK for Java 2.x 新增 APIs 实现了 HTTP /2 协议的事件流功能。有关如何使用这些新功能的示例 APIs，请参阅 [使用 Kinesis](#)。

使用来自的 SDK 指标 AWS SDK for Java

使用 AWS SDK for Java 2.x，您可以收集有关应用程序中服务客户端的指标，分析其中的输出 Amazon CloudWatch，然后对其采取行动。

默认情况下，中禁用指标收集 SDK。本主题可帮助您启用和配置指标收集。

先决条件

必须先完成以下步骤，然后才能启用并使用指标。

- 完成 [设置](#) 中的步骤。
- 将项目依赖项（例如，在您的 pom.xml 或 build.gradle 文件中）配置为使用 AWS SDK for Java 版本 2.14.0 或更高版本。

要启用向发布指标 CloudWatch，还要在项目的依赖项中加入 artifactId cloudwatch-metric-publisher 带有版本号 2.14.0 或更高版本号的。

例如：

```
<project>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>2.14.0</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>cloudwatch-metric-publisher</artifactId>
      <version>2.14.0</version>
    </dependency>
  </dependencies>
</project>
```

- 为指标发布者使用的 IAM 身份启用 cloudwatch:PutMetricData 权限，以允许 Java 编写指标。SDK

如何启用指标收集

您可以在应用程序中为服务客户端或单个请求启用指标。

为特定请求启用指标

以下课程说明如何为请求启用 CloudWatch 指标发布者 Amazon DynamoDB。该代码段使用默认的指标发布者配置。

```
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

当服务客户端不再使用时，请确保您的应用程序在[MetricPublisher](#)实例close上调用。否则，可能会导致线程或文件描述符泄漏。

为特定服务客户端启用摘要指标

以下代码片段显示了如何为服务客户端启用具有默认设置的 CloudWatch 指标发布者。

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

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

```
.build();
```

自定义指标发布者

以下课程演示如何为特定服务客户端的指标发布者设置自定义配置。自定义设置包括加载特定的配置文件、指定指标发布者向其发送请求的 AWS 区域，以及自定义发布者向其发送指标的频率。

CloudWatch

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.metrics.CoreMetric;
import software.amazon.awssdk.metrics.MetricPublisher;
import software.amazon.awssdk.metrics.publishers.cloudwatch.CloudWatchMetricPublisher;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchAsyncClient;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;

import java.time.Duration;

public class CustomConfigForDDBClient {
    // Use one MetricPublisher for your application. It can be used with requests or
    // service clients.
    static MetricPublisher metricsPub = CloudWatchMetricPublisher.builder()
        .cloudWatchClient(CloudWatchAsyncClient.builder()
            .region(Region.US_WEST_2)
            .credentialsProvider(ProfileCredentialsProvider.create("cloudwatch"))
            .build())
        .uploadFrequency(Duration.ofMinutes(5))
        .maximumCallsPerUpload(100)
        .namespace("ExampleSDKV2Metrics")
        .detailedMetrics(CoreMetric.API_CALL_DURATION)
        .build();

    public static void main(String[] args) {
        DynamoDbClient ddb = DynamoDbClient.builder()
            .overrideConfiguration(c -> c.addMetricPublisher(metricsPub))
            .build();
        // Publish metrics for DynamoDB operations.
        ddb.listTables();
        ddb.describeEndpoints();
        ddb.describeLimits();
        // Perform more work in your application.
    }
}
```

```
// A MetricsPublisher has its own lifecycle independent of any service client
or request that uses it.
// If you no longer need the publisher, close it to free up resources.
metricsPub.close(); // All metrics stored in memory are flushed to CloudWatch.

// Perform more work with the DynamoDbClient instance without publishing
metrics.
// Close the service client when you no longer need it.
ddb.close();
}
}
```

上一个片段中显示的自定义设置具有以下效果。

- 该`cloudWatchClient`方法允许您自定义用于发送指标的 CloudWatch 客户端。在此示例中，我们使用了与客户端发送指标的默认区域 `us-east-1` 不同的区域。我们还使用不同的命名配置文件 `cloudwatch`，其凭据将用于对请求进行 CloudWatch 身份验证。这些证书必须具有权限 `cloudwatch:PutMetricData`。
- 该`uploadFrequency`方法允许您指定指标发布者上传指标的频率。CloudWatch 默认值为每分钟一次。
- 该`maximumCallsPerUpload`方法限制每次上传的调用次数。默认为无限制。
- 默认情况下，for SDK Java 2.x 在命名空间 `AwsSdk/JavaSdk2` 下发布指标。您可以使用该 `namespace` 方法来指定不同的值。
- 默认情况下，会 SDK 发布汇总指标。摘要指标包括平均值、最小值、最大值、总和和样本数。通过在 `detailedMetrics` 方法中指定一个或多个 SDK 指标，可以为每个指标 SDK 发布其他数据。这些附加数据支持百分位数统计信息，例如 `p90` 和 `p99`，供您查询。CloudWatch 详细指标对于延迟指标特别有用 `APICallDuration`，例如衡量 SDK 客户端请求的 `end-to-end` 延迟。您可以使用 [CoreMetric](#) 类的字段来指定其他常用 SDK 指标。

指标何时可用？

在 for Java 发布指标后，通常会 SDK 在 5-10 分钟内提供这些指标。要获得准确性和 `up-to-date` 指标，请在从 Java 应用程序发出指标至少 10 分钟后查看 Cloudwatch。

收集哪些信息？

指标收集包括以下内容：

- API请求数量，包括请求成功还是失败
- 有关您在API请求中调用的 AWS 服务的信息，包括返回的异常
- 编组、签名和HTTP请求等各种操作的持续时间
- HTTP客户端指标，例如打开的连接数、待处理的请求数以及所用HTTP客户端的名称

Note

可用的指标因HTTP客户而异。

有关完整列表，请参阅[服务客户端指标](#)。

我该如何使用这些信息？

您可以使用SDK收集的指标来监控应用程序中的服务客户端。您可以查看总体使用趋势，识别异常情况，查看返回的服务客户端异常，或者深入了解特定问题。您还可以使用 Amazon CloudWatch创建警报，以便在应用程序达到您定义的条件时立即通知您。

有关更多信息，请参阅[Amazon CloudWatch 用户指南](#)中的[使用 Amazon CloudWatch 指标和使用 Amazon CloudWatch 警报](#)。

服务客户端指标

借助 AWS SDK for Java 2.x，您可以从应用程序中的服务客户端收集指标，然后将这些指标发布（输出）到 [Amazon CloudWatch](#)。

这些表列出了您可以收集的指标以及任何HTTP客户端使用要求。

有关为启用和配置指标的更多信息SDK，请参阅[启用SDK指标](#)。

每次请求收集的指标

指标名称	描述	类型
ApiCallDuration	完成请求所花费的总时间（包括所有重试次数）。	持续时间*
ApiCallSuccessful	如果API呼叫成功则为 true；如果不成功则为 false。	布尔值

指标名称	描述	类型
CredentialsFetchDuration	获取请求的 AWS 签名凭据所花费的时间。	持续时间*
EndpointResolveDuration	解析用于API呼叫的端点所花费的时间。	持续时间*
MarshallingDuration	将请求编组到SDK请求所花费的HTTP时间。	持续时间*
OperationName	请求 AWS API的名称已发送给。	String
RetryCount	SDK重试API呼叫的次数。	整数
ServiceId	API请求所针对 AWS 服务的服务 ID。	String
TokenFetchDuration	获取请求的令牌签名凭证所花费的时间。	持续时间*

* [java.time](#). Duration。

为每次请求尝试收集的指标

每个API呼叫可能需要多次尝试才能收到响应。每次尝试都会收集这些指标。

核心指标

指标名称	描述	类型
AwsExtendedRequestId	服务请求的扩展请求 ID。	String
AwsRequestId	服务请求的请求 ID。	String
BackoffDelayDuration	在这次API呼叫尝试之前SDK等待的时间长度。	持续时间*
ErrorType	尝试呼叫时发生的错误类型。	String

指标名称	描述	类型
ReadThroughput	客户端的读取吞吐量，以字节/秒为单位。	Double
ServiceCallDuration	连接到服务、发送请求以及从响应中接收HTTP状态码和标头所花费的时间。	持续时间*
SigningDuration	签署HTTP请求所需的时间。	持续时间*
TimeToFirstByte	从发送HTTP请求（包括获取连接）到收到响应中标头的第一个字节所经过的时间。	持续时间*
TimeToLastByte	从发送HTTP请求（包括获取连接）到收到响应的最后一个字节所经过的时间。	持续时间*
UnmarshallingDuration	取消对响应的HTTP响应所花费的SDK时间。	持续时间*

* [java.time](#). Duration。

HTTP指标

指标名称	描述	类型	HTTP需要客户*
AvailableConcurrency	无需建立其他连接即可由HTTP客户端支持的剩余并发请求数。	整数	Apache、Netty、CRT
ConcurrencyAcquireDuration	从连接池中获取频道所花费的时间。	持续时间*	Apache、Netty、CRT
HttpClientName	用于请求HTTP的名称。	String	Apache、Netty、CRT

指标名称	描述	类型	HTTP需要客户*
HttpStatusCode	HTTP响应中返回的状态码。	整数	任何
LeasedConcurrency	HTTP客户端当前正在执行的请求数。	整数	Apache、Netty、CRT
LocalStreamWindowSize	执行此请求的流的本地 HTTP /2 窗口大小 (以字节为单位)。	整数	Netty
MaxConcurrency	HTTP客户端支持的最大并发请求数。	整数	Apache、Netty、CRT
PendingConcurrencyAcquires	在连接池中等待另一个TCP连接或新数据流可用而被屏蔽的请求数。	整数	Apache、Netty、CRT
RemoteStreamWindowSize	执行此请求的流的远程 HTTP /2 窗口大小 (以字节为单位)。	整数	Netty

* [java.time](#). Duration。

该栏中使用的术语意味着：

- Apache：基于 Apache HTTP 客户端的客户端 ([ApacheHttpClient](#))
- Netty：基于 Netty 的客户端 HTTP ([NettyNioAsyncHttpClient](#))
- CRT: AWS CRT 基于的HTTP客户端 ([AwsCrtAsyncHttpClient](#))
- 任意：指标数据的收集不依赖于HTTP客户端；这包括URLConnection基于HTTP客户端的客户端 ([URLConnectionHttpClient](#))

AWS 服务 使用使用 AWS SDK for Java 2.x

本节提供有关如何使用 `select` 的简短教程和指导 AWS 服务。有关完整的示例集，请参阅“[代码示例](#)”部分。

主题

- [使用 CloudWatch](#)
- [AWS 数据库服务和 AWS SDK for Java 2.x](#)
- [与... 一起工作 DynamoDB](#)
- [与... 一起工作 Amazon EC2](#)
- [使用 IAM](#)
- [与... 一起工作 Kinesis](#)
- [调用、列出和删除 AWS Lambda 函数](#)
- [使用 Amazon S3](#)
- [使用 Amazon Simple Notification Service](#)
- [与... 一起工作 Amazon Simple Queue Service](#)
- [使用 Amazon Transcribe](#)

使用 CloudWatch

本部分提供使用 AWS SDK for Java 2.x 对 [Amazon CloudWatch](#) 进行编程的示例。

Amazon CloudWatch 实时监控您的 Amazon Web Services (AWS) 资源以及您在 AWS 中运行的应用程序。您可以使用 CloudWatch 收集和跟踪指标，这些指标是您可衡量的相关资源和应用程序的变量。CloudWatch 警报可根据您定义的规则发送通知或者对您所监控的资源自动进行更改。

以下示例仅包含演示每种方法所需的代码。[完整的示例代码在 GitHub 上提供](#)。您可以从中下载单个源文件，也可以将存储库复制到本地以获得所有示例，然后构建并运行它们。

主题

- [从 CloudWatch 中获取指标](#)
- [将自定义指标数据发布到 CloudWatch](#)
- [使用 CloudWatch 警报](#)
- [使用 Amazon CloudWatch 活动](#)

从 CloudWatch 中获取指标

列出指标

要列出 CloudWatch 指标，请创建 [ListMetricsRequest](#) 并调用 CloudWatchClient's `listMetrics` 方法。您可以使用 `ListMetricsRequest` 通过命名空间、指标名称或维度筛选返回的指标。

Note

《Amazon CloudWatch 用户指南》中的 [Amazon CloudWatch 指标和维度参考](#) 中提供了 AWS 服务发布的指标和维度列表。

导入

```
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;
```

代码

```
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);
}
}
```

[ListMetricsResponse](#)通过调用其getMetrics方法返回指标。

结果可以分页。要检索下一批结果，请对响应对象调用 nextToken 并使用该令牌值构建新的请求对象。然后使用新请求再次调用 listMetrics 方法。

请参阅上的[完整示例](#) GitHub。

更多信息

- [ListMetrics](#)在 Amazon CloudWatch API 参考中

将自定义指标数据发布到 CloudWatch

许多 AWS 服务以“AWS”开头的命名空间发布[它们自己的指标](#)。您也可以使用自己的命名空间发布自定义指标数据（不以“AWS”开头即可）。

发布自定义指标数据

要发布您自己的指标数据，请使用调用 `CloudWatchClient`'s `putMetricData` 方法 [PutMetricDataRequest](#)。 `PutMetricDataRequest` 必须包括用于数据的自定义命名空间，以及有关 [MetricDatum](#) 对象中数据点本身的信息。

Note

您无法指定以“AWS”开头的命名空间。以“AWS”开头的命名空间保留供 Amazon Web Services 产品使用。

导入

```
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;
```

代码

```
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);
}
```

请参阅上的[完整示例](#) GitHub。

更多信息

- [使用Amazon CloudWatch用户指南中的Amazon CloudWatch指标](#)。
- 《Amazon CloudWatch 用户指南》中的[AWS 命名空间](#)。
- [PutMetricData](#)在 Amazon CloudWatch API 参考中。

使用 CloudWatch 警报

创建警报

要根据CloudWatch指标创建警报，请调用[PutMetricAlarmRequest](#)填充警报条件 CloudWatchClient的's putMetricAlarm 方法。

导入

```
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;
```

代码

```
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);
    }
}
```

请参阅上的[完整示例](#) GitHub。

列出警报

要列出您创建 CloudWatchClient 的 CloudWatch 警报，请使用可以用来设置结果选项的 `describeAlarms` 方法。[DescribeAlarmsRequest](#)

导入

```
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;
```

代码

```
public static void desCWAAlarms( 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");
}
```

可以通过调用 `MetricAlarms` 返回的来获取警报列表 `describeAlarms`。 [DescribeAlarmsResponse](#)

结果可以分页。要检索下一批结果，请对响应对象调用 `nextToken` 并使用该令牌值构建新的请求对象。然后使用新请求再次调用 `describeAlarms` 方法。

Note

您还可以使用的 `describeAlarmsForMetric` 方法检索特定指标 `CloudWatchClient` 的警报。它的使用类似于 `describeAlarms`。

请参阅上的 [完整示例](#) GitHub。

删除警报

要删除 `CloudWatch` 警报，请使用 [DeleteAlarmsRequest](#) 包含一个或多个要删除的警报名称的调用 `deleteAlarms` 方法。 `CloudWatchClient`

导入

```
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;
```

代码

```
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);
}
}
```

请参阅上的[完整示例](#) GitHub。

更多信息

- [使用Amazon CloudWatch用户指南中的Amazon CloudWatch警报](#)
- [PutMetricAlarm](#)在 Amazon CloudWatch API 参考中
- [DescribeAlarms](#)在 Amazon CloudWatch API 参考中
- [DeleteAlarms](#)在 Amazon CloudWatch API 参考中

使用 Amazon CloudWatch 活动

CloudWatch 事件提供近乎实时的系统事件流，这些事件描述了 Amazon EC2 实例、Lambda 函数、Kinesis 流、Amazon ECS 任务、Step Functions 状态机、Amazon SNS 主题、Amazon SQS 队列或内置目标的 AWS 资源变化。通过使用简单的规则，您可以匹配事件并将事件路由到一个或多个目标函数或流。

Amazon EventBridge 是 CloudWatch 活动的[演变](#)。这两项服务使用相同的 API，因此您可以继续使用 SDK 提供的[CloudWatch 事件客户端](#)，也可以迁移到适用于 Java 的 SDK CloudWatch 的事件[EventBridge 客户端](#)功能。CloudWatch 活动[用户指南文档](#)和 [API 参考](#)现在可通过 EventBridge 文档网站获得。

添加事件

要添加自定义 CloudWatch 事件，请使用一个[PutEventsRequest](#)对象调用该 `CloudWatchEventsClient` 的 `sputEvents` 方法，该对象包含一个或多个提供有关每个事件的详细

信息的 [PutEventsRequestEntry](#) 对象。您可以为条目指定多个参数，例如事件的来源和类型、与事件相关联的资源等等。

Note

对于每个 `putEvents` 调用，您最多可以指定 10 个事件。

导入

```
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;
```

代码

```
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);
    }
}
```

```
}
```

请参阅上的[完整示例](#) GitHub。

添加规则

要创建或更新规则，请使用[PutRuleRequest](#)具有规则名称和可选参数（例如[事件模式](#)、要与规则关联的 IAM 角色以及描述规则运行频率的[调度表达式](#)）来调用 `CloudWatchEventsClient` 的 `putRule` 方法。

导入

```
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;
```

代码

```
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",
            ruleArn, response.ruleArn());
    } catch (
        CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

请参阅上的[完整示例](#) GitHub。

添加目标

目标是触发规则时调用的资源。示例目标包括 Amazon EC2 实例、Lambda 函数、Kinesis 流、Amazon ECS 任务、Step Functions 状态机和内置目标。

要向规则中添加目标，请使用[PutTargetsRequest](#)包含要更新的规则 and 要添加到规则中的目标列表来调用 `CloudWatchEventsClient` 的 `putTargets` 方法。

导入

```
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;
```

代码

```
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);
    }
}
```

```
}
```

请参阅上的[完整示例](#) GitHub。

更多信息

- 在 Amazon EventBridge 用户指南 PutEvents中使用@@ [添加事件](#)
- 在 Amazon EventBridge 用户指南中@@ [安排规则表达式](#)
- Amazon EventBridge 用户指南 CloudWatch Events中的@@ [事件类型](#)
- Amazon EventBridge 用户指南中的@@ [事件模式](#)
- [PutEvents](#)在 Amazon EventBridge API 参考中
- [PutTargets](#)在 Amazon EventBridge API 参考中
- [PutRule](#)在 Amazon EventBridge API 参考中

AWS 数据库服务和 AWS SDK for Java 2.x

AWS [提供了多种数据库类型：关系数据库、键值数据库、内存数据库、文档数据库和其他几种数据库](#)。适用于 Java 的 SDK 2.x 支持因中 AWS数据库服务的性质而异。

某些数据库服务（例如 [Amazon DynamoDB](#) 服务）具有用于管理 AWS 资源（数据库）的网络服务 API 以及用于与数据交互的网络服务 API。在适用于 Java 的 SDK 2.x 中，这些类型的服务有专用的服务客户端，例如 [DynamoDBClient](#)。

其他数据库服务具有与资源交互的 Web 服务 API，例如 [Amazon DocumentDB](#) API（用于集群、实例和资源管理），但没有用于处理数据的 Web 服务 API。适用于 Java 的 SDK 2.x 具有用于处理资源的相应[DocDbClient](#)接口。但是，您需要另一个 Java API，比如[适用于 Java 的 MongoDB](#) 来处理数据。

请使用以下示例来了解如何将适用于 Java 的 SDK 2.x 的服务客户端与不同类型的数据库配合使用。

Amazon DynamoDB 示例

处理数据

SDK 服务客户端：[DynamoDbClient](#)

示例：使用 DynamoDB 的 [React/Spring REST 应用程序](#)

使用数据库

SDK 服务客户端：[DynamoDbClient](#)

示例：[CreateTable](#)、[ListTables](#)、[DeleteTable](#)

处理数据

示例：[几个 DynamoDB 示例](#)

SDK 服务客户端：[DynamoDbEnhancedClient](#)

示例：使用 DynamoDB 的 [React/Spring REST 应用程序](#)

示例：[几个 DynamoDB 示例](#)（名称以“增强”开头）

使用数据库

请在本指南的指导性代码示例部分查看[其他 DynamoDB 示例](#)。

Amazon RDS 示例

处理数据	使用数据库
非 SDK API：JDBC，特定于数据库的 SQL 风格；您的代码管理数据库连接或连接池。	SDK 服务客户端： RdsClient
示例： 使用 MySQL 的 React/Spring REST 应用程序	示例： 几个 RdsClient 例子

Amazon Redshift 示例

处理数据	使用数据库
SDK 服务客户端： RedshiftDataClient	SDK 服务客户端： RedshiftClient
示例： 几个 RedshiftDataClient 例子	示例： 几个 RedshiftClient 例子
示例：使用的 React/Spring REST 应用程序 RedshiftDataClient	

亚马逊 Aurora Serverless v2 示例

处理数据	使用数据库
SDK 服务客户端： RdsDataClient	SDK 服务客户端： RdsClient
示例：使用的 React/Spring REST 应用程序 RdsDataClient	示例： 几个 RdsClient 例子

Amazon DocumentDB 示例

处理数据	使用数据库
非 SDK API：特定于 MongoDB 的 Java 库（例如 MongoDB for Java ）；您的代码管理数据库连接或连接池。	SDK 服务客户端： DocDbClient
示例： DocumentDB (Mongo) Developer Guide （选择“Java”标签）	

与... 一起工作 DynamoDB

本部分提供演示如何与 [DynamoDB](#) 结合使用的示例：

以下示例使用 2.x 的标准低级 DynamoDB 客户端 [DynamoDbClient](#) ()。AWS SDK for Java

- [the section called “使用中的表格 DynamoDB”](#)
- [the section called “处理 DynamoDB 中的项目”](#)

该 SDK 还提供了 [DynamoDB 增强型客户端](#)，为使用 DynamoDB 提供了一种面向对象的高级别方法。下一节将深入讨论此客户端。

- [the section called “将对象映射到 DynamoDB 项目”](#)

使用中的表格 DynamoDB

表是 DynamoDB 数据库中所有项目的容器。必须先创建一个表 DynamoDB，然后才能从中添加或删除数据。

对于每个表，您必须定义：

- 您的账户和地区唯一的表名。
- 一个主键，每个值对于它都必须是唯一的；表中的任意两个项目不能具有相同的主键值。

主键可以是简单主键（包含单个分区 (HASH) 键）或复合主键（包含一个分区和一个排序 (RANGE) 键）。

每个键值都有一个关联的数据类型，由该 [ScalarAttributeType](#) 类枚举。键值可以是二进制 (B)、数字 (N) 或字符串 (S)。有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的 [命名规则和数据类型](#)。

- 预置吞吐量 是定义为表保留的读取/写入容量单位数的值。

Note

[Amazon DynamoDB 定价](#) 基于您在表上设置的预配置吞吐量值，因此请仅根据您认为的表所需的容量预留容量。

表的预置吞吐量可随时修改，以便您能够在需要更改时调整容量。

创建表

使用 `DynamoDbClient.createTable` 方法创建新 DynamoDB 表。您需要构造表属性和表架构，二者用于标识表的主键。您还必须提供初始预置吞吐量值和表名。

Note

如果使用您选择的名称的表已经存在，[DynamoDbException](#) 则会抛出。

创建具有简单主键的表

此代码创建了一个表，该表的属性为表的简单主键。该示例使用了[AttributeDefinition](#)和[KeySchemaElement](#)对象。[CreateTableRequest](#)

导入

```
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;
```

代码

```
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 "";
}
```

请参阅上的[完整示例](#) GitHub。

创建具有复合主键的表

以下示例创建带有两个属性的表。这两个属性均用于复合主键。

导入

```
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;
```

代码

```
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 "";
}
```

请参阅上的[完整示例](#) GitHub。

列出表

您可以通过调用 `DynamoDbClient` 的 `listTables` 方法列出特定区域中的表。

Note

如果您的账户和区域的命名表不存在，[ResourceNotFoundException](#) 则会抛出。

导入

```
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;
```

代码

```
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);
                }
            }
        } catch (DynamoDbException e) {
            // Handle exception
        }
    }
}
```

```
        }
    } 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!");
}
```

默认情况下，每次调用最多返回 100 个表，在返回的 [ListTablesResponse](#) 对象 `lastEvaluatedTableName` 上使用来获取最后一个被评估的表。可使用此值在上一列出的最后一个返回值后开始列出。

请参阅上的 [完整示例](#) GitHub。

描述表 (获取相关信息)

使用 `DynamoDbClient` 的 `describeTable` 方法获取有关表的信息。

Note

如果您的账户和区域的命名表不存在，[ResourceNotFoundException](#) 则会抛出 a。

导入

```
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;
```

代码

```
public static void describeDynamoDBTable(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!");
}
```

请参阅上的[完整示例](#) GitHub。

修改 (更新) 表

您可以通过调用 `DynamoDbClient` 的 `updateTable` 方法随时修改表的预置吞吐量值。

Note

如果您的账户和区域的命名表不存在，[ResourceNotFoundException](#) 则会抛出 a。

导入

```
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;
```

代码

```
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!");
}
```

请参阅上的[完整示例](#) GitHub。

删除表

要删除表，请调用 `DynamoDbClient` 的 `deleteTable` 方法并提供表名称。

Note

如果您的账户和区域的命名表不存在，[ResourceNotFoundException](#) 则会抛出 a。

导入

```
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;
```

代码

```
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!");
}
```

请参阅上的[完整示例](#) GitHub。

更多信息

- 《Amazon DynamoDB 开发人员指南》中的[表处理准则](#)
- [在《Amazon DynamoDB 开发人员指南》DynamoDB中使用表](#)

处理 DynamoDB 中的项目

在 DynamoDB 中，项目是属性的集合，每个项目都包括一个名称和一个值。属性值可以为标量、集或文档类型。有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的[命名规则和数据类型](#)。

检索 (获取) 表中的项目

调用 `DynamoDbClient`'s `getItem` 方法并向其传递一个包含所需项目的表名和主键值的 `GetItemRequest` 对象。它返回一个包含该项目所有属性的 `GetItemResponse` 对象。您可以在 [中指定一个或多个投影表达式](#) `GetItemRequest` 以检索特定属性。

您可以使用返回 `GetItemResponse` 对象的 `item()` 方法来检索与该项目关联的键 (字符串 [AttributeValue](#)) 和值 () 对的[映射](#)。

导入

```
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;
```

代码

```
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);
    }
}
```

请参阅上的[完整示例](#) GitHub。

使用异步客户端从表中检索 (获取) 项目

调用getItem的方法，DynamoDbAsyncClient 然后向其传递一个包含所需项目的表名和主键值的[GetItemRequest](#)对象。

您可以返回包含该项目的所有属性的 [Collection](#) 实例 (请参阅以下示例) 。

导入

```
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;
```

代码

```
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);
    }
}
```

请参阅上的[完整示例](#) GitHub。

向表添加新项目

创建表示项目属性的键值对的[映射](#)。其中必须包括表的主键字段的值。如果主键标识的项目已存在，那么其字段将通过该请求更新。

Note

如果您的账户和地区的命名表不存在，[ResourceNotFoundException](#)则会抛出 a。

导入

```
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;
```

代码

```
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);
}
}
```

请参阅上的[完整示例](#) GitHub。

更新表中现有项目

可以使用 `DynamoDbClient` 的 `updateItem` 方法，通过提供要更新的表名称、主键值和字段映射，更新表中已有项目的属性。

Note

如果您的账户和地区的命名表不存在，或者您传入的主键标识的项目不存在，则会抛出 a [ResourceNotFoundException](#)。

导入

```
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;
```

代码

```
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!");
}
```

请参阅上的[完整示例](#) GitHub。

删除表中现有项目

您可以通过使用 `DynamoDbClient`'s `deleteItem` 方法并提供表名和主键值来删除表中存在的项目。

Note

如果您的账户和地区的命名表不存在，或者您传入的主键标识的项目不存在，则会抛出 a [ResourceNotFoundException](#)。

导入

```
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;
```

代码

```
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);
}
}
```

请参阅上的[完整示例](#) GitHub。

更多信息

- 《Amazon DynamoDB 开发人员指南》中的[项目处理准则](#)
- 《Amazon DynamoDB 开发人员指南》中的[处理 DynamoDB 中的项目](#)

使用 AWS SDK for Java 2.x 将 Java 对象映射到 DynamoDB 项目

[DynamoDB 增强型客户端 API](#) 是一个高级别库，是适用于 Java 的 SDK v1.x 中 DynamoDBMapper 类的后继库。它提供一种将客户端类映射到 DynamoDB 表的简单方法。你可以在代码中定义表与其对应的数据类之间的关系。在定义这些关系后，您可以直观地对 DynamoDB 中的表或项目执行各种创建、读取、更新或删除 (CRUD) 操作。

DynamoDB 增强型客户端 API 还包括[增强型文档 API](#)，使您能够处理不遵循已定义架构的文档类型项目。

以下主题将讨论 DynamoDB 增强型客户端 API。

- [开始使用 DynamoDB 增强型客户端 API](#)
- [了解 DynamoDB 增强型客户端 API 的基础知识](#)
- [使用高级映射功能](#)
- [使用适用于 DynamoDB 的增强型文档 API 处理 JSON 文档](#)
- [使用扩展](#)
- [异步使用 DynamoDB 增强型客户端 API](#)

- [数据类注释](#)

开始使用 DynamoDB 增强型客户端 API

以下教程向您介绍了使用 DynamoDB 增强型客户端 API 所需的基础知识。

添加依赖项

要开始在项目中使用 DynamoDB 增强型客户端 API，请添加 Maven 构件 dynamodb-enhanced 的依赖项。如以下示例所示。

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>
```

在 Maven Central 存储库中搜索 [latest version](#)，并将 `<VERSION>` 替换为该版本的值。

Gradle

```
repositories {
    mavenCentral()
}
dependencies {
    implementation(platform("software.amazon.awssdk:bom:<VERSION>"))
}
```

```
implementation("software.amazon.awssdk:dynamodb-enhanced")
...
}
```

在 Maven Central 存储库中搜索 [latest version](#)，并将 `<VERSION>` 替换为该版本的值。

TableSchema 从数据类生成

[TableSchema](#) 使增强型客户端能够将 DynamoDB 属性值映射到您的客户端类，反之亦然。在本教程中，您将了解两类 TableSchema，一类是从静态数据类派生的，另一类是使用生成器从代码中生成的。

使用带注释的数据类

适用于 Java 的 SDK 2.x 包含 [一组注释](#)，您可以将其与数据类结合使用，以快速生成 TableSchema 来将类映射到表。

首先创建一个符合 [JavaBean 规范](#) 的数据类。该规范要求类具有无参数的公共构造函数，并且类中的每个属性都有 getter 和 setter。包括类级别的注释，以指示数据类是 DynamoDbBean。此外，至少要在 getter 或 setter 上包含关于主键属性的 DynamoDbPartitionKey 注释。

您可以将 [属性级注释](#) 应用于 getter 或 setter，但不能同时应用两者。

Note

该术语 property 通常用于封装在 a 中的值。JavaBean 但是，为了与 DynamoDB 使用的术语保持一致，本指南（英文版）改用术语 attribute。（中文版中，两者的翻译均为“属性”。）

以下 Customer 类显示了将类定义链接到 DynamoDB 表的注释。

Customer 类

```
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 + "]";
    }
}
```

创建带注释的数据类后，使用它来创建 `TableSchema`，如以下代码段所示。

```
static final TableSchema<Customer> customerTableSchema =
    TableSchema.fromBean(Customer.class);
```

`TableSchema` 被设计为静态且不可变。您通常可以在类加载时将其实例化。

静态 `TableSchema.fromBean()` 工厂方法对 Bean 进行内省，生成数据类属性（属性）与 DynamoDB 属性的映射。

有关使用由多个数据类组成的数据模型的示例，请参阅[???部分](#)中的 `Person` 类。

使用生成器

如果您在代码中定义表架构，则可以避免 Bean 自检的开销。如果您对架构进行编码，则您的类无需遵循 JavaBean 命名标准，也不需要对其进行注释。以下示例使用生成器，与使用注释的 `Customer` 类示例等效。

```
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();
```

创建增强型客户端和 `DynamoDbTable`

创建增强型客户端

该[DynamoDbEnhancedClient](#)类或其异步类是使用 DynamoDB 增强型客户端 API 的入口点。[DynamoDbEnhancedAsyncClient](#)

增强型客户端需要标准 [DynamoDbClient](#) 才能执行工作。API 提供了两种创建 `DynamoDbEnhancedClient` 实例的方法。第一个选项是使用从配置设置中选取的默认设置创建标准 `DynamoDbClient`，如以下代码段所示。

```
DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.create();
```

如果要配置底层标准客户端，可以将其提供给增强型客户端的生成器方法，如以下代码段所示。

```
// 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();
```

创建 **DynamoDbTable** 实例

可以将 [DynamoDbTable](#) 视为使用 TableSchema 提供的映射功能的 DynamoDB 表的客户端表示形式。该 DynamoDbTable 类提供了 CRUD 操作的方法，允许您与单个 DynamoDB 表进行交互。

DynamoDbTable<T> 是一个采用单一类型参数的通用类，无论是自定义类还是处理文档类型项目时的 EnhancedDocument。此参数类型在您使用的类和单个 DynamoDB 表之间建立关系。

使用 DynamoDbEnhancedClient 的 table() 工厂方法创建 DynamoDbTable 实例，如以下代码段所示。

```
static final DynamoDbTable<Customer> customerTable =
    enhancedClient.table("Customer", TableSchema.fromBean(Customer.class));
```

DynamoDbTable 实例是单例类的候选实例，因为它们是不可变的，可以在整个应用程序中使用。

现在，您的代码具有可以处理实例的 DynamoDB 表的内存表示形式。Customer 实际的 DynamoDB 表可能存在，也可能不存在。如果名为 Customer 的表已经存在，则可以开始对其执行 CRUD 操作。如果该表不存在，请使用 DynamoDbTable 实例创建表，如下一部分所述。

如果需要，创建 DynamoDB 表

创建 DynamoDbTable 实例后，使用它在 DynamoDB 中一次性创建表。

创建表示例代码

以下示例基于 Customer 数据类创建一个 DynamoDB 表。

此示例创建了一个 DynamoDB 表，其名称为 Customer（与类名称相同，但表名称可以是其他名称）。不管您如何为表命名，都必须在其他应用程序中使用该名称才能使用该表。为了使用底层 DynamoDB 表，每次创建另一个 DynamoDbTable 对象时，都要为 table() 方法提供此名称。

传递给 `createTable` 方法的 Java lambda 参数 `builder` 允许您[自定义表](#)。在此示例中，配置了[预置吞吐量](#)。要在创建表时使用默认设置，请跳过生成器，如以下代码段所示。

```
customerTable.createTable();
```

使用默认设置时，不会设置预置吞吐量的值。取而代之的是，表的计费模式将设置为[按需](#)。

该示例还在尝试打印出响应中收到的表名称之前使用了 [DynamoDbWaiter](#)。创建表需要一些时间。因此，使用 `waiter` 意味着您不必编写在使用表之前轮询 DynamoDB 服务，以查看表是否存在的逻辑。

导入

```
import com.example.dynamodb.Customer;
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbEnhancedClient;
import software.amazon.awssdk.enhanced.dynamodb.DynamoDbTable;
import software.amazon.awssdk.enhanced.dynamodb.TableSchema;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;
```

代码

```
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

从数据类生成表时，DynamoDB 表的属性名称以小写字母开头。如果您希望表的属性名称以大写字母开头，请使用 [@DynamoDbAttribute\(NAME\)](#) 注释并提供所需的名称作为参数。

执行操作

创建表后，请使用 `DynamoDbTable` 实例对 DynamoDB 表执行操作。

在以下示例中，将单例 `DynamoDbTable<Customer>` 作为参数与 [Customer 数据类](#) 实例一起传递，以向表中添加新项目。

```
public static void putItemExample(DynamoDbTable<Customer> customerTable, Customer
customer){
    logger.info(customer.toString());
    customerTable.putItem(customer);
}
```

Customer 对象

```
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"));
```

在将 `customer` 对象发送到 DynamoDB 服务之前，请记录对象的 `toString()` 方法的输出，以将其与增强型客户端发送的内容进行比较。

```
Customer [id=1, name=Customer Name, email=customer@example.com,  
regDate=2023-07-03T10:15:30Z]
```

线程级日志记录显示生成的请求的有效负载。增强型客户端从数据类生成了低级别表示形式。regDate 属性是 Java 中的一种 Instant 类型，以 DynamoDB 字符串的形式表示。

```
{  
  "TableName": "Customer",  
  "Item": {  
    "registrationDate": {  
      "S": "2023-07-03T10:15:30Z"  
    },  
    "id": {  
      "S": "1"  
    },  
    "custName": {  
      "S": "Customer Name"  
    },  
    "email": {  
      "S": "customer@example.com"  
    }  
  }  
}
```

使用现有表

上一部分介绍了如何从 Java 数据类开始，创建 DynamoDB 表。如果您已有表，并且想要使用增强型客户端的功能，则可以创建一个 Java 数据类来使用该表。您需要检查 DynamoDB 表并为数据类添加必要的注释。

在使用现有表之前，请调用 `DynamoDbEnhanced.table()` 方法。在前面的示例中，这是通过以下语句完成的。

```
DynamoDbTable<Customer> customerTable = enhancedClient.table("Customer",  
TableSchema.fromBean(Customer.class));
```

返回 `DynamoDbTable` 实例后，您可以立即开始使用底层表。您无需通过调用 `DynamoDbTable.createTable()` 方法来重新创建表。

以下示例通过立即从 DynamoDB 表中检索 `Customer` 实例，演示了这一点。

```
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()` 方法中使用的表名称必须与现有 DynamoDB 表名称相匹配。

了解 DynamoDB 增强型客户端 API 的基础知识

本主题讨论 DynamoDB 增强型客户端 API 的基本功能，并将其与[标准 DynamoDB 客户端 API](#)进行了比较。

如果您不熟悉 DynamoDB 增强型客户端 API，建议您阅读[介绍性教程](#)，熟悉基础类。

Java 中的 DynamoDB 项目

DynamoDB 表用于存储项目。根据您的用例，Java 端的项目可以采用静态结构化数据形式或动态创建结构形式。

如果您的用例要求项目使用一组一致的属性，请使用[带注释的类](#)或使用[生成器](#)生成相应的静态类型 `TableSchema`。

或者，如果您需要存储不同结构的项目，请创建一个

`DocumentTableSchema`。`DocumentTableSchema` 是[增强型文档 API](#)的一部分，只需要静态类型的主键，并且可以与 `EnhancedDocument` 实例结合使用来保存数据元素。增强型文档 API 将在另一个[主题](#)中介绍。

数据模型类的属性类型

尽管与 Java 丰富的类型系统相比，DynamoDB 支持的[属性类型较少](#)，但 DynamoDB 增强型客户端 API 提供了将 Java 类的成员与 DynamoDB 属性类型相互转换的机制。

Java 数据类的属性类型（属性）应该是对象类型，而不是基元。例如，始终使用 `Long` 和 `Integer` 对象数据类型，而不是 `long` 和 `int` 基元。

默认情况下，DynamoDB 增强型客户端 API 支持大量类型的属性转换器，例如整数 `BigDecimal`、字符串和即时。该列表显示在 [AttributeConverter 接口的已知实现类中](#)。该列表包括许多类型和集合，例如映射、列表和集。

要存储默认不支持或不符合同约定的属性类型的数据，可以编写自定义 `AttributeConverter` 实现来进行转换。JavaBean 有关 [示例](#)，请参阅属性转换部分。

要存储属性类型的类符合 Java Bean 规范（或该类为 [不可变数据类](#)）的数据，可以采用两种方法。

- 如果您有权访问源文件，则可以使用 `@DynamoDbBean`（或 `@DynamoDbImmutable`）为该类添加注释。讨论嵌套属性的部分提供了使用带注释的类的 [示例](#)。
- 如果无权访问该属性的 JavaBean 数据类的源文件（或者您不想为自己有权访问的类的源文件添加注释），则可以使用生成器方法。这可在不定义键的情况下创建表架构。然后，您可以将此表架构嵌套在另一个表架构中以执行映射。嵌套属性部分提供了使用嵌套架构的 [示例](#)。

Null 值

使用 `putItem` API 时，增强型客户端不会在向 DynamoDB 发出的请求中包含映射数据对象的空值属性。

对于 `updateItem` 请求，将从数据库的项目中移除空值属性。如果您打算更新某些属性值并保持其他属性值不变，请复制不应更改的其他属性的值，或者使用更新生成器上的 [ignoreNull\(\)](#) 方法。

以下示例演示 the `updateItem()` 方法的 `ignoreNulls()`。

```
public void updateItemNullsExample(){
    Customer customer = new Customer();
    customer.setCustName("CustName");
    customer.setEmail("email");
    customer.setId("1");
    customer.setRegistrationDate(Instant.now());

    // Put item with values for all attributes.
    customerDynamoDbTable.putItem(customer);

    // Create a Customer instance with the same id value, but a different name
    value.
    // Do not set the 'registrationDate' attribute.
    Customer custForUpdate = new Customer();
    custForUpdate.setCustName("NewName");
```

```

    custForUpdate.setEmail("email");
    custForUpdate.setId("1");

    // Update item without setting the registrationDate attribute.
    customerDynamoDbTable.updateItem(b -> b
        .item(custForUpdate)
        .ignoreNulls(Boolean.TRUE));

    Customer updatedWithNullsIgnored = customerDynamoDbTable.getItem(customer);
    // registrationDate value is unchanged.
    logger.info(updatedWithNullsIgnored.toString());

    customerDynamoDbTable.updateItem(custForUpdate);
    Customer updatedWithNulls = customerDynamoDbTable.getItem(customer);
    // registrationDate value is null because ignoreNulls() was not used.
    logger.info(updatedWithNulls.toString());
}
}

// Logged lines.
Customer [id=1, custName=NewName, email=email,
    registrationDate=2023-04-05T16:32:32.056Z]
Customer [id=1, custName=NewName, email=email, registrationDate=null]

```

DynamoDB 增强型客户端基本方法

增强型客户端的基本方法映射到它们以之命名的 DynamoDB 服务操作。以下示例演示每种方法的最简单变体。您可以通过传入增强型请求对象来自定义每种方法。增强型请求对象提供了标准 DynamoDB 客户端中可用的大部分功能。它们完整记录在《AWS SDK for Java 2.x API Reference》中。

该示例使用了前面所示的 [the section called “Customer 类”](#)。

```

// 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));
```

比较 DynamoDB 增强型客户端与标准 DynamoDB 客户端

这两个 DynamoDB 客户端 API ([标准](#)和[增强型](#)) 都允许您使用 DynamoDB 表来执行数据级 CRUD (创建、读取、更新和删除) 操作。客户端 API 之间的区别在于该操作是如何实现的。使用标准客户端，您可以直接处理低级别数据属性。增强型客户端 API 使用熟悉的 Java 类，并映射到后台的低级别 API。

虽然两个客户端 API 都支持数据级操作，但标准 DynamoDB 客户端也支持资源级操作。资源级操作管理数据库，例如创建备份、列出表和更新表。增强型客户端 API 支持一定数量的资源级操作，例如创建、描述和删除表。

为了说明两个客户端 API 使用的不同方法，以下代码示例演示如何使用标准客户端和增强型客户端创建同一个 ProductCatalog 表。

比较：使用标准 DynamoDB 客户端创建表

```
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))
);
```

比较：使用 DynamoDB 增强型客户端创建表

```
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))
    );
```

增强型客户端使用以下带注释的数据类。DynamoDB 增强型客户端将 Java 数据类型映射到 DynamoDB 数据类型，代码不那么冗长，更易于理解。ProductCatalog 是在 DynamoDB 增强型客户端中使用不可变类的一个例子。本主题[后面将讨论](#)为映射的数据类使用不可变类。

ProductCatalog 类

```
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);
        }
    }
}

```

以下两个批量写入代码示例说明了使用标准客户端时，相较于增强型客户端，代码的冗长和缺乏类型安全性。

比较：使用标准 DynamoDB 客户端的批量写入操作

```

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());
}

```

比较：使用 DynamoDB 增强型客户端的批量写入操作

```

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());
}

```

使用不可变数据类

DynamoDB 增强型客户端 API 的映射功能适用于不可变的数据类。不可变类只有 getter，且需要一个生成器类，使 SDK 可用来创建该类的实例。不可变类不像 [Customer 类](#) 那样使用 `@DynamoDbBean` 注释，而是使用 `@DynamoDbImmutable` 注释，该注释采用一个指示要使用的生成器类的参数。

以下类是 `Customer` 的不可变版本。

```

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); }
}
}

```

使用 `@DynamoDbImmutable` 注释数据类时，您必须满足以下要求。

1. 每个既不覆盖 `Object.class` 又未使用 `@DynamoDbIgnore` 注释的方法都必须是 DynamoDB 表属性的 getter。
2. 每个 getter 在生成器类上都必须有一个对应的区分大小写的 setter。
3. 必须仅满足以下任一构造条件。
 - 生成器类必须具有公共默认构造函数。
 - 数据类必须有一个名为 `builder()` 的公共静态方法，该方法不带任何参数并返回生成器类的实例。此选项显示在不可变 `Customer` 类中。
4. 生成器类必须有一个名为 `build()` 的公共方法，该方法不带任何参数并返回不可变类的实例。

要为不可变类创建 `TableSchema`，请使用 `TableSchema` 上的 `fromImmutableClass()` 方法，如下代码段所示。

```

static final TableSchema<CustomerImmutable> customerImmutableTableSchema =
    TableSchema.fromImmutableClass(CustomerImmutable.class);

```

就像您可以从可变类创建 DynamoDB 表一样，您也可以通过一次性调用 `DynamoDbTable` 的 `createTable()` 从不可变类创建该表，如下代码段示例所示。

```

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.waitForTableExists(b -> b.tableName(tableName));
}

```

使用第三方库，例如 Lombok

第三方库（例如 [Project Lombok](#)）可帮助生成与不可变对象关联的样板代码。只要数据类遵循本部分详述的约定，DynamoDB 增强型客户端 API 就可以与这些库结合使用。

以下示例演示带有 Lombok 注释的不可变 `CustomerImmutable` 类。请注意 Lombok 的 `onMethod` 功能是如何将基于属性的 DynamoDB 注释（例如 `@DynamoDbPartitionKey`）复制到生成的代码中的。

```

@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 createdAt;
}

```

使用表达式和条件

DynamoDB 增强型客户端 API 中的表达式是 [DynamoDB 表达式](#) 的 Java 表示形式。

DynamoDB 增强型客户端 API 使用三种类型的表达式：

[Expression](#)

Expression 类在定义条件和筛选条件时使用。

[QueryConditional](#)

这种类型的表达式表示查询操作的[关键条件](#)。

[UpdateExpression](#)

该类可帮助您编写 DynamoDB [更新](#)表达式，当您更新项目时，目前会在扩展框架中使用该类。

表达式刨析

表达式由以下内容组成：

- 字符串表达式（必需）。该字符串包含一个 DynamoDB 逻辑表达式，其中包含属性名称和属性值的占位符名称。
- 表达式值映射（通常是必需的）。
- 表达式名称映射（可选）。

使用生成器生成一个采用以下一般形式的 Expression 对象。

```
Expression expression = Expression.builder()
    .expression(<String>)
    .expressionNames(<Map>)
    .expressionValues(<Map>)
    .build()
```

Expression 通常需要表达式值映射。映射提供了字符串表达式中占位符的值。映射键由前面带有冒号(:)的占位符名称组成，映射值是 [AttributeValue](#) 的实例。[AttributeValues](#) 类具有从文字生成 AttributeValue 实例的便捷方法。或者，您可以使用 AttributeValue.Builder 生成 AttributeValue 实例。

以下代码段展示了在注释行 2 之后有两个条目的映射。传递给 expression() 方法的字符串（显示在注释行 1 之后）包含 DynamoDB 在执行操作之前解析的占位符。此代码段不包含表达式名称映射，因为 price 是允许的属性名称。

```
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();

```

如果 DynamoDB 表中的属性名称是保留字、以数字开头或包含空格，则 Expression 需要表达式名称映射。

例如，如果属性名称是 `1price`，而不是前面的代码示例中的 `price`，则需要修改示例，如以下示例所示。

```

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();

```

表达式名称的占位符以井号 (#) 开头。表达式名称映射的条目使用占位符作为键，使用属性名称作为值。使用 `expressionNames()` 方法将映射添加到表达式生成器中。DynamoDB 会在执行操作之前解析属性名称。

如果在字符串表达式中使用函数，则不需要表达式值。表达式函数的一个例子是 `attribute_exists(<attribute_name>)`。

以下示例构建了一个使用 [DynamoDB 函数](#) 的 Expression。本示例中的表达式字符串不使用占位符。此表达式可用于 `putItem` 操作以检查数据库中是否已存在 `movie` 属性值等于数据对象的 `movie` 属性的项目。

```
Expression exp = Expression.builder().expression("attribute_not_exists  
(movie)").build();
```

《DynamoDB 开发人员指南》包含有关可用于 DynamoDB 的[低级别表达式](#)的完整信息。

条件表达式和条件语句

使用 `putItem()`、`updateItem()` 和 `deleteItem()` 方法时，以及使用事务和批处理操作时，您可以使用 [Expression](#) 对象来指定 DynamoDB 必须满足什么条件才能继续执行操作。这些表达式是命名条件表达式。有关示例，请参阅本指南中显示的[事务示例](#)的 `addDeleteItem()` 方法中使用的条件表达式（在注释行 1 之后）。

使用 `query()` 方法时，条件表示为 [QueryConditional](#)。QueryConditional 类有几种静态便利方法，可帮助您编写用于确定要从 DynamoDB 读取哪些项目的标准。

有关 QueryConditionals 的示例，请参阅本指南[the section called “Query 方法示例”](#)部分的第一个代码示例。

筛选条件表达式

筛选表达式用于扫描和查询操作以筛选返回的项目。

从数据库中读取所有数据后，会应用筛选表达式，因此读取成本与不使用筛选条件时的读取成本相同。《Amazon DynamoDB 开发人员指南》提供了有关使用筛选表达式进行[查询](#)和[扫描](#)操作的更多信息。

以下示例展示了添加到扫描请求的筛选表达式。该标准将返回的项目限制为价格介于 8.00 到 80.00（含）之间的项目。

```
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();
```

更新表达式

DynamoDB 增强型客户端的 `updateItem()` 方法提供了一种在 DynamoDB 中更新项目的标准方法。但是，当您需要更多功能时，[UpdateExpressions](#) 会提供 DynamoDB [更新表达式语法](#) 的类型安全表示形式。例如，您可以使用 `UpdateExpressions` 增加值而不必先读取 DynamoDB 中的项目，或者将单个成员添加到列表中。更新表达式目前在 `updateItem()` 方法的自定义扩展中可用。

有关使用更新表达式的示例，请参阅本指南中的[自定义扩展示例](#)。

有关更新表达式的更多信息，请参阅《[Amazon DynamoDB 开发人员指南](#)》。

处理分页结果：扫描和查询

DynamoDB 增强型客户端 API 的 `scan`、`query` 和 `batch` 方法返回包含一个或多个页面的响应。一个页面包含一个或多个项目。您的代码可以按页处理响应，也可以处理单个项目。

同步 `DynamoDbEnhancedClient` 客户端返回的分页响应返回一个 [PageIterable](#) 对象，而异步客户端返回的响应 `DynamoDbEnhancedAsyncClient` 返回一个 [PagePublisher](#) 对象。

本部分介绍如何处理分页结果，并提供使用扫描和查询 API 的示例。

扫描表

SDK 的 [scan](#) 方法对应于同名的 [DynamoDB 操作](#)。DynamoDB 增强型客户端 API 提供了相同的选项，但它使用熟悉的对象模型并为您处理分页。

首先，我们通过查看同步映射类的 `scan` 方法来探索 `PageIterable` 接口 [DynamoDbTable](#)。

使用同步 API

以下示例演示使用 [表达式](#) 筛选返回项的 `scan` 方法。[ProductCatalog](#) 是前面显示的模型对象。

在评论行 2 之后显示的筛选表达式将退回的 `ProductCatalog` 商品限制为价格在 8.00 到 80.00 之间（含）的商品。

此示例还使用注释行 1 后面显示的 `attributesToProject` 方法排除这些 `isbn` 值。

在注释第 3 行之后 `pagedResults`，`scan` 方法返回 `PageIterable` 对象。`PageIterable` 的 `stream` 方法返回一个 [java.util.Stream](#) 对象，您可以使用该对象来处理页面。在此示例中，计算并记录了页数。

从注释行 4 开始，该示例演示访问 ProductCatalog 项目的两种变体。注释行 4a 之后的版本通过每个页面进行流式传输，并对每页上的项目进行排序和记录。注释行 4b 之后的版本会跳过页面迭代并直接访问项目。

由于 PageIterable 接口有两个父接口 ([java.lang.Iterable](#) 和 [SdkIterable](#))，因此提供了多种处理结果的方。Iterable 引入了 forEach、iterator 和 spliterator 方法，而 SdkIterable 引入了 stream 方法。

```
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())
        );
}
```

```
    );  
}
```

使用异步 API

异步 `scan` 方法将结果作为 `PagePublisher` 对象返回。`PagePublisher` 接口有两种 `subscribe` 方法可用于处理响应页面。一种 `subscribe` 方法来自 `org.reactivestreams.Publisher` 父接口。要使用第一个选项处理页面，请向 `subscribe` 方法传递一个 [Subscriber](#) 实例。接下来的第一个示例演示了 `subscribe` 方法的用法。

第二种 `subscribe` 方法来自 [SdkPublisher](#) 接口。此版本的 `subscribe` 接受 [Consumer](#) 而不是 `Subscriber`。此 `subscribe` 方法变体如接下来的第二个示例所示。

以下示例演示 `scan` 方法的异步版本，该版本使用了上一个示例中的相同筛选表达式。

在注释行 3 之后，`DynamoDbAsyncTable.scan` 返回一个 `PagePublisher` 对象。在下一行，代码创建一个 `org.reactivestreams.Subscriber` 接口实例 `ProductCatalogSubscriber`，在注释行 4 之后，该实例订阅到 `PagePublisher`。

在 `ProductCatalogSubscriber` 类示例的注释行 8 之后，`Subscriber` 对象从 `onNext` 方法中的每个页面收集 `ProductCatalog` 项目。这些项目存储在私有 `List` 变量中，并在调用代码中使用 `ProductCatalogSubscriber.getSubscribedItems()` 方法对它们进行访问。这是在注释行 5 之后调用的。

检索了列表后，代码按价格对所有 `ProductCatalog` 项目进行排序并记录每个项目。

`ProductCatalogSubscriber` 类 [CountDownLatch](#) 中的会阻塞调用线程，直到所有项目都已添加到列表中，然后在注释行 5 之后继续。

```
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;
    }
}

```

以下代码段示例使用的 `PagePublisher.subscribe` 方法版本在注释行 6 之后接受 `Consumer`。Java lambda 参数使用页面，进一步处理每个项目。在此示例中，对每个页面进行处理，并对每页上的项目进行排序和记录。

```

// 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.
```

`PagePublisher` 的 `items` 方法对模型实例进行解包，以便您的代码可以直接处理这些项目。以下代码段演示了这种方法。

```
// 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.
```

查询表

`DynamoDbTable` 类的 [query\(\)](#) 方法基于主键值查找项目。`@DynamoDbPartitionKey` 注释和可选的 `@DynamoDbSortKey` 注释用于定义数据类的主键。

`query()` 方法需要一个分区键值来查找与提供的值相匹配的项目。如果您的表还定义了排序键，则可以在查询中为它添加一个值作为额外的比较条件来微调结果。

除了处理结果之外，同步版本和异步版本 `query()` 的工作原理相同。与 `scan` API 一样，`query` API 会为同步调用返回 `PageIterable`，为异步调用返回 `PagePublisher`。我们之前在扫描部分讨论了 `PageIterable` 和 `PagePublisher` 的使用。

Query 方法示例

下面的 `query()` 方法代码示例使用 `MovieActor` 类。数据类定义了一个复合主键，该主键由分区键的 `movie` 属性和排序键的 `actor` 属性组成。

该类还表示它使用名为 `acting_award_year` 的全局二级索引。索引的复合主键由分区键的 `actingaward` 属性和排序键的 `actingyear` 属性组成。在本主题的后面部分，我们将在演示如何创建和使用索引时，引用 `acting_award_year` 索引。

MovieActor 类

```
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);
        }
    }
}
```

对以下项目进行查询之后的代码示例。

MovieActor 表中的项目

```
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'}

```

以下代码定义了两个[QueryConditional](#)实例。QueryConditionals使用键值（可以单独使用分区键，也可以与排序键组合使用），并与 DynamoDB 服务 API 的[密钥条件表达式](#)相对应。在注释行 1 之后，该示例定义了与分区值为 **movie01** 的项目匹配的 `keyEqual` 实例。

此示例还在注释行 2 之后，定义了一个筛选表达式，过滤掉任何没有 **actingschoolname** 的项目。

在注释第 3 行之后，该示例显示了代码传递给 `DynamoDbTable.query()` 方法的[QueryEnhancedRequest](#)实例。此对象结合了 SDK 用来生成对 DynamoDB 服务的请求的关键条件和筛选条件。

```

public static void query(DynamoDbTable movieActorTable) {

    // 1. Define a QueryConditional instance to return items matching a partition
    value.
    QueryConditional keyEqual = QueryConditional.keyEqualTo(b ->
    b.partitionValue("movie01"));
    // 1a. Define a QueryConditional that adds a sort key criteria to the partition
    value criteria.
    QueryConditional sortGreaterThanOrEqualTo =
    QueryConditional.sortGreaterThanOrEqualTo(b ->
    b.partitionValue("movie01").sortValue("actor2"));
    // 2. Define a filter expression that filters out items whose attribute value
    is null.
    final Expression filterOutNoActingschoolname =
    Expression.builder().expression("attribute_exists(actingschoolname)").build();

    // 3. Build the query request.
    QueryEnhancedRequest tableQuery = QueryEnhancedRequest.builder()
        .queryConditional(keyEqual)
        .filterExpression(filterOutNoActingschoolname)
        .build();
    // 4. Perform the query.

```

```

PageIterable<MovieActor> pagedResults = movieActorTable.query(tableQuery);
logger.info("page count: {}", pagedResults.stream().count()); // Log number of
pages.

pagedResults.items().stream()
    .sorted()
    .forEach(
        item -> logger.info(item.toString()) // Log the sorted list of
items.
    );

```

下面是运行该方法的输出。该输出显示 `movieName` 值为 `movie01` 的项目，不显示 `actingSchoolName` 等于 `null` 的项目。

```

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'}

```

在以下查询请求代码变体中，之前在注释行 3 之后显示的 `keyEqual QueryConditional` 将替换为注释行 1a 之后定义的 `sortGreaterThanOrEqualTo QueryConditional`。以下代码还删除了筛选表达式。

```

QueryEnhancedRequest tableQuery = QueryEnhancedRequest.builder()
    .queryConditional(sortGreaterThanOrEqualTo)

```

由于此表具有复合主键，因此所有 `QueryConditional` 实例都需要分区键值。以 `sort...` 开头的 `QueryConditional` 方法指示需要排序键。结果未排序。

以下输出显示了查询的结果。该查询仅返回 `movieName` 值等于 `movie01` 且 `actorName` 值大于或等于 `actor2` 的项目。由于筛选条件已被删除，因此查询会返回没有 `actingSchoolName` 属性值的项目。

```

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'}
```

执行批量操作

DynamoDB 增强型客户端 API 提供两种批处理方法：[batchGetItem\(\)](#) 和 [batchWriteItem\(\)](#)。

batchGetItem() 示例

使用 [DynamoDbEnhancedClient.batchGetItem\(\)](#) 方法，您可以在一个总请求中检索多个表中最多 100 个单独的项目。以下示例使用前面显示的 [Customer](#) 和 [MovieActor](#) 数据类。

在示例的注释行 1 和 2 之后，您将构建 [ReadBatch](#) 对象，然后在注释行 3 之后将其作为参数添加到 [batchGetItem\(\)](#) 方法中。注释行 1 之后的代码生成要从 [Customer](#) 表中读取的批次。注释行 1a 之后的代码显示了如何使用 [GetItemEnhancedRequest](#) 生成器，该生成器采用主键值来指定要读取的项目。与指定键值来请求项目不同，您可以使用数据类来请求项目，如注释行 1b 后所示。提交请求之前，SDK 会在后台提取键值。

如 2a 之后的两个语句所示，当您使用基于键的方法指定项目时，您还可以指定 DynamoDB 应执行[强一致性读取](#)。使用 [consistentRead\(\)](#) 方法时，必须对同一个表的所有请求项目使用该方法。

要检索 DynamoDB 找到的项目，请使用注释行 4 之后显示的 [resultsForTable\(\)](#) 方法。为请求中读取的每个表调用该方法。[resultsForTable\(\)](#) 返回可使用任何 `java.util.List` 方法处理的已找到项目的列表。此示例记录每个项目。

要发现 DynamoDB 未处理的项目，请使用注释行 5 之后的方法。[BatchGetResultPage](#) 类具有允许您访问每个未处理键的 [unprocessedKeysForTable\(\)](#) 方法。[BatchGetItem API 参考](#)提供了有关导致项目未处理的情况的更多信息。

```
public static void batchGetItemExample(DynamoDbEnhancedClient enhancedClient,
                                       DynamoDbTable<Customer> customerTable,
                                       DynamoDbTable<MovieActor> movieActorTable) {

    Customer customer2 = new Customer();
    customer2.setId("2");
```

```
customer2.setEmail("cust2@example.org");

// 1. Build a batch to read from the Customer table.
ReadBatch customerBatch = ReadBatch.builder(Customer.class)
    .mappedTableResource(customerTable)
    // 1a. Specify the primary key values for the item.
    .addGetItem(b -> b.key(k ->
k.partitionValue("1").sortValue("cust1@orgname.org")))
    // 1b. Alternatively, supply a data class instances to provide the
primary key values.
    .addGetItem(customer2)
    .build();

// 2. Build a batch to read from the MovieActor table.
ReadBatch moveActorBatch = ReadBatch.builder(MovieActor.class)
    .mappedTableResource(movieActorTable)
    // 2a. Call consistentRead(Boolean.TRUE) for each item for the same
table.
    .addGetItem(b -> b.key(k ->
k.partitionValue("movie01").sortValue("actor1")))
    .consistentRead(Boolean.TRUE))
    .addGetItem(b -> b.key(k ->
k.partitionValue("movie01").sortValue("actor4")))
    .consistentRead(Boolean.TRUE))
    .build();

// 3. Add ReadBatch objects to the request.
BatchGetResultPageIterable resultPages = enhancedClient.batchGetItem(b ->
b.readBatches(customerBatch, moveActorBatch));

// 4. Retrieve the successfully requested items from each table.
resultPages.resultsForTable(customerTable).forEach(item ->
logger.info(item.toString()));
resultPages.resultsForTable(movieActorTable).forEach(item ->
logger.info(item.toString()));

// 5. Retrieve the keys of the items requested but not processed by the
service.
resultPages.forEach((BatchGetResultPage pageResult) -> {
    pageResult.unprocessedKeysForTable(customerTable).forEach(key ->
logger.info("Unprocessed item key: " + key.toString()));
    pageResult.unprocessedKeysForTable(movieActorTable).forEach(key ->
logger.info("Unprocessed item key: " + key.toString()));
});
}
```

在运行示例代码之前，假设两个表中包含以下项目。

表格中的项目

```
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'}
```

以下输出显示了在注释行 4 之后返回和记录的项目。

```
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() 示例

batchWriteItem() 方法在一个或多个表中放置或删除多个项目。您最多可以在请求中指定 25 个单独的放置或删除操作。以下示例使用前面显示的 [ProductCatalog](#) 和 [MovieActor](#) 模型类。

WriteBatch 对象是在注释行 1 和 2 之后生成的。对于 ProductCatalog 表，代码放置一个项目并删除一个项目。对于注释行 2 之后的 MovieActor 表，代码放置了两个项目并删除了一个项目。

在注释行 3 之后调用 `batchWriteItem` 方法。[builder](#) 参数提供每个表的批处理请求。

返回的 [BatchWriteResult](#) 对象为每个操作提供了不同的方法来查看未处理的请求。注释行 4a 之后的代码为未处理的删除请求提供了键，注释行 4b 之后的代码提供了未处理的放置项目。

```
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()));
        }
    }
}
```

以下帮助程序方法为放置和删除操作提供了模型对象。

帮助程序方法

```
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;
}

```

在运行示例代码之前，假设这些表包含以下项目。

```

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}

```

示例代码完成后，表中将包含以下项目。

```

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}

```

请注意，在 `MovieActor` 表中，`Blue Jasmine` 电影项目已被替换为通过 `getMovieActorBlanchettPartial()` 帮助程序方法获取的放置请求中使用的项。如果未提供数据 Bean 属性值，则数据库中的值将被删除。这就是为什么 `Blue Jasmine` 影片项目的结果 `actingSchoolName` 为空的原因。

Note

尽管 API 文档表示可以使用条件表达式，并且可以在单独的[放置](#)和[删除](#)请求中返回已消耗的容量和集合指标，但在批量写入场景中，情况并非如此。为了提高批处理操作的性能，将忽略这些单独的选项。

执行事务操作

DynamoDB 增强型客户端 API 提供了 `transactGetItems()` 和 `transactWriteItems()` 方法。适用于 Java 的 SDK 的事务方法在 DynamoDB 表中提供原子性、一致性、隔离性和持久性 (ACID)，帮助您维护应用程序中的数据正确性。

`transactGetItems()` 示例

[transactGetItems\(\)](#) 方法最多可接受 100 个单独的项目请求。所有项目都在单个原子事务中读取。《Amazon DynamoDB 开发人员指南》包含有关[导致 transactGetItems\(\) 方法失败的条件](#)以及您调用 [transactGetItem\(\)](#) 时使用的隔离级别的信息。

在以下示例的注释行 1 之后，代码使用 [builder](#) 参数调用 `transactGetItems()` 方法。使用一个数据对象调用生成器的 [addGetItem\(\)](#) 三次，该数据对象包含 SDK 将用于生成最终请求的键值。

该请求在注释行 2 之后返回 [Document](#) 对象列表。返回的文档列表包含项目数据的非空 [Document](#) 实例，其顺序与请求的顺序相同。如果返回了项目数据，则 [Document.getItem\(MappedTableResource<T> mappedTableResource\)](#) 方法会将非类型化 `Document` 对象转换为类型化的 Java 对象，否则该方法返回 `null`。

```
public static void transactGetItemsExample(DynamoDbEnhancedClient enhancedClient,
                                           DynamoDbTable<ProductCatalog>
catalogTable,
                                           DynamoDbTable<MovieActor>
movieActorTable) {

    // 1. Request three items from two tables using a builder.
    final List<Document> documents = enhancedClient.transactGetItems(b -> b
        .addGetItem(catalogTable,
Key.builder().partitionValue(2).sortValue("Title 55").build())
        .addGetItem(movieActorTable, Key.builder().partitionValue("Sophie's
Choice").sortValue("Meryl Streep").build())
        .addGetItem(movieActorTable, Key.builder().partitionValue("Blue
Jasmine").sortValue("Cate Blanchett").build())
        .build());

    // 2. A list of Document objects is returned in the same order as requested.
    ProductCatalog title55 = documents.get(0).getItem(catalogTable);
    if (title55 != null) {
        logger.info(title55.toString());
    }
}
```

```

    MovieActor sophiesChoice = documents.get(1).getItem(movieActorTable);
    if (sophiesChoice != null) {
        logger.info(sophiesChoice.toString());
    }

    // 3. The getItem() method returns null if the Document object contains no item
    from DynamoDB.
    MovieActor blueJasmine = documents.get(2).getItem(movieActorTable);
    if (blueJasmine != null) {
        logger.info(blueJasmine.toString());
    }
}

```

在代码示例运行之前，DynamoDB 表包含以下项目。

```

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'}

```

记录了以下输出。如果请求了某个项目但未找到，则该项目不会返回（像请求名为 Blue Jasmine 的电影那样）。

```

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() 示例

[transactWriteItems\(\)](#) 在跨多个表的单个原子事务中最多接受 100 个放置、更新或删除操作。

《Amazon DynamoDB 开发人员指南》包含有关[底层 DynamoDB 服务操作](#)的限制和失败条件的详细信息。

基本示例

在以下示例中，请求对两个表执行四个操作。之前已显示了相应的模型类 [ProductCatalog](#) 和 [MovieActor](#)。

三种可能的操作（放置、更新和删除）均使用专用的请求参数来指定详细信息。

注释行 1 之后的代码显示了 `addPutItem()` 方法的简单变体。该方法接受要放置的 [MappedTableResource](#) 对象和数据对象实例。注释行 2 之后的语句显示了接受

[TransactPutItemEnhancedRequest](#) 实例的变体。此变体允许您在请求中添加更多选项，例如条件表达式。随后的[示例](#)显示了单个操作的条件表达式。

在注释行 3 之后请求更新操作。[TransactUpdateItemEnhancedRequest](#) 有一种 `ignoreNulls()` 方法可以让您配置 SDK 如何处理模型对象上的 `null` 值。如果 `ignoreNulls()` 方法返回 `true`，则对于数据对象属性为 `null` 的情况，SDK 不会移除表的属性值。如果 `ignoreNulls()` 方法返回 `false`，则 SDK 会请求 DynamoDB 服务从表中的项目中移除这些属性。`ignoreNulls` 的默认值为 `false`。

注释行 4 之后的语句显示了接受数据对象的删除请求的变体。增强型客户端在分派最终请求之前提取键值。

```
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())

    );
}
```

以下帮助程序方法为 `add*Item` 参数提供了数据对象。

帮助程序方法

```
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;
}
```

在代码示例运行之前，DynamoDB 表包含以下项目。

```
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'}
```

代码运行完毕后，表中将显示以下项目。

```
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'}
```

第 2 行的项目已被删除，第 3 行和第 5 行显示了已放置的项目。第 4 行显示第 1 行的更新。price 值是该项目上唯一更改的值。如果 ignoreNulls() 返回 false，第 4 行将类似于以下行。

```
ProductCatalog{id=4, title='Title 1', isbn='null', authors=null, price=40.0}
```

条件检查示例

以下示例演示条件检查的使用。条件检查用于检查项目是否存在，或者检查数据库中项目特定属性的条件。条件检查中检查的项目不能用于事务中的其他操作。

Note

您不能将同一个事务中的多个操作指向同一个项目。例如，您不能在同一个事务中对相同项目既执行条件检查又执行更新操作。

该示例演示事务写入项目请求中的每种操作类型。在注释行 2 之后，如果 conditionExpression 参数的计算结果为 false，则 addConditionCheck() 方法会提供事务失败的条件。从帮助程序方法块中显示的方法返回的条件表达式会检查电影 Sophie's Choice 的获奖年份是否不等于 1982。如果是，则表达式的计算结果为 false，事务将失败。

本指南的另一个主题深入讨论了[表达式](#)。

```
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());
            });
        }
    }
}

```

前面的代码示例中使用了以下帮助程序方法。

帮助程序方法

```

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;
}

```

在代码示例运行之前，DynamoDB 表包含以下项目。

- 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'}

代码运行完毕后，表中将显示以下项目。

```
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'}
```

由于事务失败，表中的项目保持不变。影片 Sophie's Choice 的 actingYear 值为 1982，如调用 transactWriteItem() 方法之前表中项目的第 2 行所示。

要捕获事务的取消信息，请将 transactWriteItems() 方法调用封装在一个 try 块中，然后 catch [TransactionCanceledException](#)。在示例的注释行 4 之后，代码会记录每个 [CancellationReason](#) 对象。由于示例注释行 3 之后的代码指定应返回导致事务失败的项目的值，因此日志会显示电影项目 Sophie's Choice 的原始数据库值。

```
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.)
```

单一操作条件示例

以下示例演示在事务请求中对单个操作使用条件的情况。注释行 1 之后的删除操作包含一个条件，该条件用于根据数据库检查操作的目标项目的值。在此示例中，在注释行 2 之后使用帮助程序方法创建的条件表达式指定，如果电影的上映年份不等于 2013 年，则应从数据库中删除该项目。

本指南稍后将讨论[表达式](#)。

```
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();
}
}

```

前面的代码示例中使用了以下帮助程序方法。

帮助程序方法

```

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;
    }
}

```

在代码示例运行之前，DynamoDB 表包含以下项目。

```

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'}

```

代码运行完毕后，表中将显示以下项目。

```

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'}

```

由于事务失败，表中的项目保持不变。在代码示例运行之前，影片 Blue Jasmine 的 actingYear 值为 2013，如项目列表第 2 行所示。

以下行记录到控制台。

```

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)

```

使用二级索引

二级索引通过定义在查询和扫描操作中使用的备用键来改善数据访问。全局二级索引 (GSI) 的分区键和排序键可与基表不同。相比之下，本地二级索引 (LSI) 使用主索引的分区键。

使用二级索引注释对数据类进行注释

参与二级索引的属性需要使用 `@DynamoDbSecondaryPartitionKey` 或 `@DynamoDbSecondarySortKey` 注释。

以下类显示了两个索引的注释。命名的 GSI `SubjectLastPostedDateIndex` 使用该 `Subject` 属性作为分区键，使用属性 `LastPostedDateTime` 作为排序键。命名的 LSI `ForumLastPostedDateIndex` 使用 `ForumName` 作为其分区键和 `LastPostedDateTime` 排序键。

请注意，`Subject` 属性起着双重作用。它是主键的排序键，也是命名 `SubjectLastPostedDateIndex` 的 GSI 的分区键。

MessageThread 类

`MessageThread` 类适合用作《Amazon DynamoDB 开发人员指南》中 [示例 Thread 表](#) 的数据类。

导入

```

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 +
            '}';
    }
}
```

创建索引

从适用于 Java 的 SDK 的 2.20.86 版本开始，`createTable()` 方法会自动根据数据类标注生成二级索引。默认情况下，基表中的所有属性都将复制到索引中，预置吞吐量值为 20 个读取容量单位和 20 个写入容量单位。

但是，如果您使用的是 2.20.86 之前的 SDK 版本，则需要将索引与表一起构建，如下示例所示。此示例为 Thread 表构建两个索引。[builder](#) 参数具有配置两种索引的方法，如注释行 1 和 2 所示。您可以使用索引生成器的 `indexName()` 方法将数据类注释中指定的索引名称与预期的索引类型相关联。

在注释行 3 和 4 之后，此代码将所有表属性配置为最后出现在两个索引中。有关[属性投影](#)的更多信息，请参阅《Amazon DynamoDB 开发人员指南》。

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

使用索引查询

以下示例查询本地二级索引ForumLastPostedDateIndex。

在注释行 2 之后，您将创建一个调用 [DynamoDbIndex.query\(\)](#) 方法时所需的[QueryConditional](#)对象。

在注释行 3 之后，通过传入索引名称，即可获得对要查询的索引的引用。在注释行 4 之后，通过传入 [QueryConditional](#) 对象在索引上调用 [query\(\)](#) 方法。

您还可以将查询配置为返回三个属性值，如注释行 5 之后所示。如果 [attributesToProject\(\)](#) 未调用，则查询将返回所有属性值。请注意，指定的属性名称以小写字母开头。这些属性名称与表中使用的属性名称相匹配，不一定与数据类的属性名称相匹配。

在注释行 6 之后，迭代结果、记录查询返回的每个项目，并将其存储在列表中以返回给调用方。

```
public static List<MessageThread> queryUsingSecondaryIndices(DynamoDbEnhancedClient
enhancedClient,
    String lastPostedDate,
    DynamoDbTable<MessageThread> threadTable) {
    // 1. Log the parameter value.
    logger.info("lastPostedDate value: {}", lastPostedDate);
}
```

```

    // 2. Create a QueryConditional whose sort key value must be greater than or
    equal to the parameter value.
    QueryConditional queryConditional =
    QueryConditional.sortGreaterThanOrEqualTo(qc ->
        qc.partitionValue("Forum02").sortValue(lastPostedDate));

    // 3. Specify the index name to query the DynamoDbIndex instance.
    final DynamoDbIndex<MessageThread> forumLastPostedDateIndex =
    threadTable.index("ForumLastPostedDateIndex");

    // 4. Perform the query by using the QueryConditional object.
    final SdkIterable<Page<MessageThread>> pagedResult =
    forumLastPostedDateIndex.query(q -> q
        .queryConditional(queryConditional)
        // 5. Request three attribute in the results.
        .attributesToProject("forumName", "subject", "lastPostedDateTime"));

    List<MessageThread> collectedItems = new ArrayList<>();
    // 6. Iterate through the pages response and sort the items.
    pagedResult.stream().forEach(page -> page.items().stream()

    .sorted(Comparator.comparing(MessageThread::getLastPostedDateTime))
        .forEach(mt -> {
            // 7. Log the returned items and add the collection to
            return to the caller.
            logger.info(mt.toString());
            collectedItems.add(mt);
        }));
    return collectedItems;
}

```

在运行查询之前，数据库中存在以下项目。

```

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}

```

第 1 行和第 6 行的日志语句会生成以下控制台输出。

```

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}

```

该查询返回 forumName 值为 Forum02，lastPostedDateTime 值大于或等于 2023.03.31 的项目。尽管索引中的 message 属性具有值，但结果显示的 message 值为空字符串。这是因为在注释行 5 之后，未投影消息属性。

使用高级映射功能

了解 DynamoDB 增强型客户端 API 中的高级表架构功能。

了解表架构类型

[TableSchema](#) 是 DynamoDB 增强型客户端 API 映射功能的接口。它可以将数据对象映射到地图或从地图映射出来 [AttributeValues](#)。TableSchema 对象需要知道它所映射的表的结构。此结构信息存储在 [TableMetadata](#) 对象中。

增强型客户端 API 有以下几种 TableSchema 实现。

从带注释的类生成的表架构

从带注释的类构建 TableSchema 的操作成本适中，因此我们建议在应用程序启动时执行一次。

[BeanTableSchema](#)

此实现是基于 Bean 类的属性和注释构建的。[入门部分](#) 演示了这种方法的示例。

Note

如果 BeanTableSchema 的行为不符合您的预期，请为 `software.amazon.awssdk.enhanced.dynamodb.beans` 启用调试日志记录。

[ImmutableTableSchema](#)

此实现基于不可变的数据类构建。[???部分](#) 介绍了此方法。

使用生成器生成的表架构

以下 TableSchema 是使用生成器根据代码构建的。这种方法比使用带注释的数据类的方法更便宜。构建器方法避免使用注释，并且不需要 JavaBean 命名标准。

[StaticTableSchema](#)

此实现是为可变数据类构建的。本指南的入门部分演示了如何[使用生成器生成 StaticTableSchema](#)。

[StaticImmutableTableSchema](#)

与 StaticTableSchema 构建方式类似，您使用[生成器](#)生成这种类型的 TableSchema 实现，以用于不可变的数据类。

适用于无固定架构数据的表架构

[DocumentTableSchema](#)

与其他 `TableSchema` 实现不同，您无需为 `DocumentTableSchema` 实例定义属性。通常，您只需指定主键和属性转换器提供程序。`EnhancedDocument` 实例将提供您根据单个元素或 JSON 字符串构建的属性。

明确包含或排除属性

DynamoDB 增强型客户端 API 提供了注释，可将数据类属性排除在表的属性之外。通过 API，您还可以使用与数据类属性名称不同的属性名称。

排除属性

要忽略不应映射到 DynamoDB 表的属性，请使用 `@DynamoDbIgnore` 注释标记该属性。

```
private String internalKey;

@DynamoDbIgnore
public String getInternalKey() { return this.internalKey; }
public void setInternalKey(String internalKey) { this.internalKey = internalKey;}
```

包含属性

要更改 DynamoDB 表中使用的属性的名称，请使用 `@DynamoDbAttribute` 注释标记该属性并提供其他名称。

```
private String internalKey;

@DynamoDbAttribute("renamedInternalKey")
public String getInternalKey() { return this.internalKey; }
public void setInternalKey(String internalKey) { this.internalKey = internalKey;}
```

控制属性转换

默认情况下，表架构通过 [AttributeConverterProvider](#) 接口的默认实现为许多常见的 Java 类型提供转换器。您可以使用自定义 `AttributeConverterProvider` 实现来更改整体默认行为。您还可以更改单个属性的转换器。

有关可用转换器的列表，请参阅 [AttributeConverter](#) 接口 Java 文档。

提供自定义属性转换器提供程序

您可以通过 `@DynamoDbBean (converterProviders = {...})` 注释提供单个 `AttributeConverterProvider` 或一个有序的 `AttributeConverterProvider` 链。任何自定义 `AttributeConverterProvider` 都必须扩展 `AttributeConverterProvider` 接口。

请注意，如果您提供自己的属性转换器提供程序链，则将覆盖默认的转换器提供程序 `DefaultAttributeConverterProvider`。如果您要使用 `DefaultAttributeConverterProvider` 的功能，必须将其包含在链中。

也可以用空 `{}` 数组对 `Bean` 进行注释。这将禁用任何属性转换器提供程序，包括默认提供程序。在这种情况下，所有要映射的属性都必须有自己的属性转换器。

以下代码段显示了单个转换器提供程序。

```
@DynamoDbBean(converterProviders = ConverterProvider1.class)
public class Customer {

}
```

以下代码段显示了转换器提供程序链的用法。由于 SDK 默认转换器排在最后，因此它的优先级最低。

```
@DynamoDbBean(converterProviders = {
    ConverterProvider1.class,
    ConverterProvider2.class,
    DefaultAttributeConverterProvider.class})
public class Customer {

}
```

静态表架构生成器有一种工作方式与此相同的 `attributeConverterProviders()` 方法。如以下代码段所示。

```
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();
```

覆盖单个属性的映射

要覆盖单个属性的映射方式，请为该属性提供一个 `AttributeConverter`。此添加会覆盖表架构中由 `AttributeConverterProviders` 提供的任何转换器。这将仅为该属性添加一个自定义转换器。除非为其他属性明确指定该转换器，否则其他属性即使类型相同，也不会使用该转换器。

`@DynamoDbConvertedBy` 注释用于指定自定义 `AttributeConverter` 类，如以下代码段所示。

```
@DynamoDbBean
public class Customer {
    private String name;

    @DynamoDbConvertedBy(CustomAttributeConverter.class)
    public String getName() { return this.name; }
    public void setName(String name) { this.name = name;}
}
```

静态架构的生成器具有等效的属性生成器 `attributeConverter()` 方法。此方法采用 `AttributeConverter` 的实例，如下所示。

```
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();
```

示例

此示例演示一个为 [java.net.HttpCookie](http://java.net/HttpCookie) 对象提供属性转换器的 `AttributeConverterProvider` 实现。

以下 `SimpleUser` 类包含一个名为 `lastUsedCookie` 的属性，该属性是 `HttpCookie` 的一个实例。

`@DynamoDbBean` 注释的参数列出了提供转换器的两个 `AttributeConverterProvider` 类。

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();

```

以下示例中的 `CookieConverterProvider` 提供了 `HttpCookeConverter` 的一个实例。

```

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);
    }
}

```

代码转换

在以下 `HttpCookieConverter` 类的 `transformFrom()` 方法中，代码接收一个 `HttpCookie` 实例并将其转换为 DynamoDB 映射，并将该映射作为属性存储。

`transformTo()` 方法接收 DynamoDB 映射参数，然后调用需要名称和值的 `HttpCookie` 构造函数。

```

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;
    }
}

```

更改属性的更新行为

在执行更新操作时，您可以自定义各个属性的更新行为。[DynamoDB 增强型客户端 API 中的一些更新操作示例如下 `updateItem\(\)` 和 `transactWriteItems\(\)`](#)。

例如，假设您要在记录中存储 `created on` 时间戳。但是，您希望仅在数据库中没有该属性的现有值时才写入其值。在这种情况下，您可以使用 [WRITE_IF_NOT_EXISTS](#) 更新行为。

以下示例展示了将行为添加到 `createdOn` 属性的注释。

```

@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; }
}

```

在构建静态表架构时，您可以声明相同的更新行为，如以下示例的注释行 1 之后所示。

```

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();

```

扁平化其他类的属性

如果表的属性通过继承或组合分布在多个不同的 Java 类中，则 DynamoDB 增强型客户端 API 支持将这些属性扁平化为一个类。

使用继承

如果您的类使用继承，请使用以下方法来扁平化层次结构。

使用带注释的 Bean

对于注释方法，两个类都必须带有 @DynamoDbBean 注释，并且一个类必须带有一个或多个主键注释。

以下是具有继承关系的数据类的示例。

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 createdAt;

    @DynamoDbPartitionKey
    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getCreatedAt() { return createdAt; }
    public void setCreatedAt(String createdAt) { this.createdAt =
    createdAt; }
}
```

Lombok

Lombok 的 [onMethod 选项](#) 将基于属性的 DynamoDB 注释（例如 @DynamoDbPartitionKey）复制到生成的代码中。

```
@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 createdAt;
}
```

使用静态架构

对于静态架构方法，使用生成器的 extend() 方法将父类的属性折叠到子类上。如以下示例的注释行 1 之后所示。

```
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();

```

前面的静态架构示例使用以下数据类。由于映射是在构建静态表架构时定义的，因此数据类不需要注释。

数据类

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 createdAt;

    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getCreatedAt() { return createdAt; }
    public void setCreatedAt(String createdAt) { this.createdAt =
createdAt; }
}

```

Lombok

```

@Data
@ToString(callSuper = true)
public class Customer extends GenericRecord{
    private String name;
}

@Data
public abstract class GenericRecord {
    private String id;
    private String createdAt;
}

```

使用组合

如果您的类使用组合，请使用以下方法来扁平化层次结构。

使用带注释的 Bean

使用 `@DynamoDbFlatten` 注释将所含的类扁平化。

以下数据类示例使用 `@DynamoDbFlatten` 注释将所含的 `GenericRecord` 类的所有属性有效添加到 `Customer` 类中。

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 createdAt;

        @DynamoDbPartitionKey
        public String getId() { return this.id; }
        public void setId(String id) { this.id = id; }

        public String getCreatedAt() { return this.createdAt; }
        public void setCreatedAt(String createdAt) { this.createdAt =
        createdAt; }
    }
}

```

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 createdAt;
}

```


您可以根据需要，使用扁平化注释对任意数量的符合条件的不同类进行扁平化。以下限制适用：

- 所有属性名称在扁平化后必须是唯一的。
- 分区键、排序键或表名称不得超过一个。

使用静态架构

构建静态表架构时，请使用生成器的 `flatten()` 方法。您还可以提供用于识别所含类的 `getter` 和 `setter` 方法。

```
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();
```

前面的静态架构示例使用以下数据类。

数据类

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 createdAt;

    public String getId() { return this.id; }
    public void setId(String id) { this.id = id; }

    public String getCreatedAt() { return this.createdAt; }
    public void setCreatedAt(String createdAt) { this.createdAt =
    createdAt; }
}
```

Lombok

```
@Data
public class Customer {
    private String name;
    private GenericRecord record;
}

@Data
public class GenericRecord {
    private String id;
    private String createdAt;
}
```

您可以根据需要，使用生成器模式对任意数量的符合条件的不同类进行扁平化。

对其他代码的影响

当您使用 `@DynamoDbFlatten` 属性（或 `flatten()` 生成器方法）时，DynamoDB 中的项目将针对组合成的对象的每个属性包含一个属性。它还包括进行组合的对象的属性。

相反，如果您使用组合成的类对数据类进行注释但不使用 `@DynamoDbFlatten`，则该项目将与组合成的对象一起保存为单个属性。

例如，我们可以比较[使用组合的扁平化示例](#)中显示的 `Customer` 类在对 `record` 属性进行扁平化和不进行扁平化时的区别。您可以使用 JSON 可视化此区别，如下表所示。

进行扁平化	不进行扁平化
3 个属性	2 个属性
<pre>{ "id": "1", "createdDate": "today", "name": "my name" }</pre>	<pre>{ "id": "1", "record": { "createdDate": "today", "name": "my name" } }</pre>

如果您有其他代码访问 DynamoDB 表并希望找到某些属性，则此区别就变得很重要。

处理嵌套属性

DynamoDB 中的嵌套属性嵌入在另一个属性中，例如列表元素和映射条目。

在 Java 中，DynamoDB 嵌套属性对应于 `List` 或 `Map` 类的成员。它还对应于复杂类型的实例，例如 `Address` 或 `PhoneNumber`，如用于以下 `Person` 类中。

Person 类

```
@DynamoDbBean
public class Person {
    Integer id;
    String firstName;
    String lastName;
    Integer age;
    Map<String, Address> addresses;
    List<PhoneNumber> phoneNumbers;

    List<String> hobbies;

    @DynamoDbPartitionKey
```

```
public Integer getId() {
    return id;
}

public void setId(Integer id) {
    this.id = id;
}

public String getFirstName() {
    return firstName;
}

public void setFirstName(String firstName) {
    this.firstName = firstName;
}

public String getLastName() {
    return lastName;
}

public void setLastName(String lastName) {
    this.lastName = lastName;
}

public Integer getAge() {
    return age;
}

public void setAge(Integer age) {
    this.age = age;
}

public Map<String, Address> getAddresses() {
    return addresses;
}

public void setAddresses(Map<String, Address> addresses) {
    this.addresses = addresses;
}

public List<PhoneNumber> getPhoneNumbers() {
    return phoneNumbers;
}
```

```
public void setPhoneNumbers(List<PhoneNumber> phoneNumbers) {
    this.phoneNumbers = phoneNumbers;
}

public List<String> getHobbies() {
    return hobbies;
}

public void setHobbies(List<String> hobbies) {
    this.hobbies = hobbies;
}

@Override
public String toString() {
    return "Person{" +
        "id=" + id +
        ", firstName='" + firstName + '\'' +
        ", lastName='" + lastName + '\'' +
        ", age=" + age +
        ", addresses=" + addresses +
        ", phoneNumbers=" + phoneNumbers +
        ", hobbies=" + hobbies +
        '}';
}
}
```

Address 类

```
@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 类

```
@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 + '\'' +
            '}';
    }
}
```

映射嵌套属性

使用带注释的类

您可以通过注释来保存自定义类的嵌套属性。前面显示的 `Address` 类和 `PhoneNumber` 类仅使用 `@DynamoDbBean` 注释进行注释。当 DynamoDB 增强型客户端 API 使用以下代码段为 `Person` 类生成表架构时，API 会发现 `Address` 和 `PhoneNumber` 类的使用，并生成相应的映射以与 DynamoDB 结合使用。

```
TableSchema<Person> personTableSchema = TableSchema.fromBean(Person.class);
```

使用嵌套架构

另一种方法是为每个类使用静态表架构生成器，如以下代码所示。

`Address` 和 `PhoneNumber` 类的表架构是抽象的，不能与 DynamoDB 表一起使用。这是因为它们缺少主键的定义。但是，它们在 `Person` 类的表架构中用作嵌套架构。

在注释行 1 和 2 之后的 `PERSON_TABLE_SCHEMA` 定义中，显示了使用抽象表架构的代码。在 `EnhanceType.documentOf(...)` 方法中使用 `documentOf` 并不表示该方法将返回增强型文档 API 的 `EnhancedDocument` 类型。在此上下文中，`documentOf(...)` 方法将返回一个对象，该对象知道如何使用表架构参数，在其类参数和 DynamoDB 表属性之间进行映射。

静态架构代码

```
// Abstract table schema that cannot be used to work with a DynamoDB table,  
// but can be used as a nested schema.  
public static final TableSchema<Address> TABLE_SCHEMA_ADDRESS =  
TableSchema.builder(Address.class)  
    .newItemSupplier(Address::new)  
    .addAttribute(String.class, a -> a.name("street")  
        .getter(Address::getStreet)  
        .setter(Address::setStreet))  
    .addAttribute(String.class, a -> a.name("city")  
        .getter(Address::getCity)  
        .setter(Address::setCity))  
    .addAttribute(String.class, a -> a.name("zipcode")  
        .getter(Address::getZipCode)  
        .setter(Address::setZipCode))  
    .addAttribute(String.class, a -> a.name("state")  
        .getter(Address::getState)  
        .setter(Address::setState))  
    .build();
```



```

// Abstract table schema that cannot be used to work with a DynamoDB table,
// but can be used as a nested schema.
public static final TableSchema<PhoneNumber> TABLE_SCHEMA_PHONENUMBER =
TableSchema.builder(PhoneNumber.class)
    .newItemSupplier(PhoneNumber::new)
    .addAttribute(String.class, a -> a.name("type")
        .getter(PhoneNumber::getType)
        .setter(PhoneNumber::setType))
    .addAttribute(String.class, a -> a.name("number")
        .getter(PhoneNumber::getNumber)
        .setter(PhoneNumber::setNumber))
    .build();

// A static table schema that can be used with a DynamoDB table.
// The table schema contains two nested schemas that are used to perform mapping
to/from DynamoDB.
public static final TableSchema<Person> PERSON_TABLE_SCHEMA =
    TableSchema.builder(Person.class)
        .newItemSupplier(Person::new)
        .addAttribute(Integer.class, a -> a.name("id")
            .getter(Person::getId)
            .setter(Person::setId)
            .addTag(StaticAttributeTags.primaryPartitionKey()))
        .addAttribute(String.class, a -> a.name("firstName")
            .getter(Person::getFirstName)
            .setter(Person::setFirstName))
        .addAttribute(String.class, a -> a.name("lastName")
            .getter(Person::getLastName)
            .setter(Person::setLastName))
        .addAttribute(Integer.class, a -> a.name("age")
            .getter(Person::getAge)
            .setter(Person::setAge))
        .addAttribute(EnhancedType.listOf(String.class), a ->
a.name("hobbies")
            .getter(Person::getHobbies)
            .setter(Person::setHobbies))
        .addAttribute(EnhancedType.mapOf(
            EnhancedType.of(String.class),
            // 1. Use mapping functionality of the Address table
            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();

```

投影嵌套属性

对于 `query()` 和 `scan()` 方法，您可以使用 `addNestedAttributeToProject()` 和 `attributesToProject()` 之类的方法调用来指定要在结果中返回哪些属性。在发送请求之前，DynamoDB 增强型客户端 API 会将 Java 方法调用参数转换为 [投影表达式](#)。

以下示例在 `Person` 表中填充两个项目，然后执行三个扫描操作。

第一个扫描访问表中的所有项目，以便将结果与其他扫描操作进行比较。

第二个扫描使用 [addNestedAttributeToProject\(\)](#) 生成器方法仅返回 `street` 属性值。

第三个扫描操作使用 [attributesToProject\(\)](#) 生成器方法返回第一级属性 `hobbies` 的数据。`hobbies` 的属性类型是列表。要访问单个列表项目，请对列表执行 `get()` 操作。

```

    personDynamoDbTable = getDynamoDbEnhancedClient().table("Person",
PERSON_TABLE_SCHEMA);
    PersonUtils.createPersonTable(personDynamoDbTable, getDynamoDbClient());
    // Use a utility class to add items to the Person table.
    List<Person> personList = PersonUtils.getItemsForCount(2);
    // This utility method performs a put against DynamoDB to save the instances in
the list argument.
    PersonUtils.putCollection(getDynamoDbEnhancedClient(), personList,
personDynamoDbTable);

    // The first scan logs all items in the table to compare to the results of the
subsequent scans.
    final PageIterable<Person> allItems = personDynamoDbTable.scan();
    allItems.items().forEach(p ->
        // 1. Log what is in the table.
        logger.info(p.toString()));

    // Scan for nested attributes.

```

```

    PageIterable<Person> streetScanResult = personDynamoDbTable.scan(b -> b
        // Use the 'addNestedAttributeToProject()' or
        'addNestedAttributesToProject()' to access data nested in maps in DynamoDB.
        .addNestedAttributeToProject(
            NestedAttributeName.create("addresses", "work", "street")
        ));

    streetScanResult.items().forEach(p ->
        //2. Log the results of requesting nested attributes.
        logger.info(p.toString()));

    // Scan for a top-level list attribute.
    PageIterable<Person> phoneNumbersScanResult = personDynamoDbTable.scan(b -> b
        // Use the 'attributesToProject()' method to access first-level
attributes.
        .attributesToProject("hobbies"));

    phoneNumbersScanResult.items().forEach((p) -> {
        // 3. Log the results of the request for the 'hobbies' attribute.
        logger.info(p.toString());
        // To access an item in a list, first get the parent attribute, 'hobbies',
then access items in the list.
        String hobby = p.getHobbies().get(1);
        // 4. Log an item in the list.
        logger.info(hobby);
    });

```

```

// Logged results from comment line 1.
Person{id=2, firstName='first name 2', lastName='last name 2', age=11,
addresses={work=Address{street='street 21', city='city 21', state='state 21',
zipCode='33333'}, home=Address{street='street 2', city='city 2', state='state 2',
zipCode='22222'}}, phoneNumbers=[PhoneNumber{type='home', number='222-222-2222'},
PhoneNumber{type='work', number='333-333-3333'}], hobbies=[hobby 2, hobby 21]}
Person{id=1, firstName='first name 1', lastName='last name 1', age=11,
addresses={work=Address{street='street 11', city='city 11', state='state 11',
zipCode='22222'}, home=Address{street='street 1', city='city 1', state='state 1',
zipCode='11111'}}, phoneNumbers=[PhoneNumber{type='home', number='111-111-1111'},
PhoneNumber{type='work', number='222-222-2222'}], hobbies=[hobby 1, hobby 11]}

// Logged results from comment line 2.
Person{id=null, firstName='null', lastName='null', age=null,
addresses={work=Address{street='street 21', city='null', state='null',
zipCode='null'}}}, phoneNumbers=null, hobbies=null}

```

```

Person{id=null, firstName='null', lastName='null', age=null,
  addresses={work=Address{street='street 11', city='null', state='null',
  zipCode='null'}}}, phoneNumbers=null, hobbies=null}

// Logged results from comment lines 3 and 4.
Person{id=null, firstName='null', lastName='null', age=null, addresses=null,
  phoneNumbers=null, hobbies=[hobby 2, hobby 21]}
hobby 21
Person{id=null, firstName='null', lastName='null', age=null, addresses=null,
  phoneNumbers=null, hobbies=[hobby 1, hobby 11]}
hobby 11

```

Note

如果 `attributesToProject()` 方法遵循任何其他用于添加要投影的属性的生成器方法，则提供给 `attributesToProject()` 的属性名称列表将替换所有其他属性名称。在以下代码段中，使用 `ScanEnhancedRequest` 实例执行的扫描仅返回业余爱好数据。

```

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]}

```

以下代码段首先使用 `attributesToProject()` 方法。此排序保留了请求的所有其他属性。

```

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]}

```

使用 `@DynamoDbPreserveEmptyObject` 保留空对象

如果您将包含空对象的 Bean 保存到 Amazon DynamoDB 中，并且希望 SDK 在检索时重新创建空对象，请使用 `@DynamoDbPreserveEmptyObject` 注释内部 Bean 的 getter。

为了说明该注释的工作原理，代码示例使用了以下两个 Bean。

示例 Bean

以下数据类包含两个 InnerBean 字段。getter 方法 `getInnerBeanWithoutAnno()` 不使用 `@DynamoDbPreserveEmptyObject` 注释。 `getInnerBeanWithAnno()` 方法使用注释。

```

@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();
}
}

```

以下 InnerBean 类的实例是 MyBean 的字段，并且在示例代码中被初始化为空对象。

```

@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 + '\'' +
        '}';
}
}

```

以下代码示例将带有初始化内部 Bean 的 MyBean 对象保存到 DynamoDB，然后检索该项目。记录的输出显示 innerBeanWithoutAnno 未初始化，但已创建 innerBeanWithAnno。

```

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;
}

```

替代静态架构

您可以使用以下 StaticTableSchema 版本的表架构来代替 Bean 上的注释。

```

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();
}

```

避免保存嵌套对象的空属性

在将数据类对象保存到 DynamoDB 时，您可以通过应用 `@DynamoDbIgnoreNulls` 注释来跳过嵌套对象的空属性。相比之下，具有空值的顶级属性永远不会保存到数据库中。

为了说明该注释的工作原理，代码示例使用了以下两个 Bean。

示例 Bean

以下数据类包含两个 `InnerBean` 字段。getter 方法 `getInnerBeanWithoutAnno()` 不使用注释。getter 方法 `getInnerBeanWithIgnoreNullsAnno()` 方法使用注释 `@DynamoDbIgnoreNulls`。

```

@DynamoDbBean
public class MyBean {

    private String id;
    private String name;
    private InnerBean innerBeanWithoutAnno;

```



```

private InnerBean innerBeanWithIgnoreNullsAnno;

@DynamoDbPartitionKey
public String getId() { return id; }
public void setId(String id) { this.id = id; }

public String getName() { return name; }
public void setName(String name) { this.name = name; }

public InnerBean getInnerBeanWithoutAnno() { return innerBeanWithoutAnno; }
public void setInnerBeanWithoutAnno(InnerBean innerBeanWithoutAnno)
{ this.innerBeanWithoutAnno = innerBeanWithoutAnno; }

@DynamoDbIgnoreNulls
public InnerBean getInnerBeanWithIgnoreNullsAnno() { return
innerBeanWithIgnoreNullsAnno; }
public void setInnerBeanWithIgnoreNullsAnno(InnerBean innerBeanWithAnno)
{ this.innerBeanWithIgnoreNullsAnno = innerBeanWithAnno; }

@Override
public String toString() {
    return new StringJoiner(", ", MyBean.class.getSimpleName() + "[", "]")
        .add("innerBeanWithoutAnno=" + innerBeanWithoutAnno)
        .add("innerBeanWithIgnoreNullsAnno=" + innerBeanWithIgnoreNullsAnno)
        .add("id='" + id + "'")
        .add("name='" + name + "'")
        .toString();
}
}

```

以下 InnerBean 类的实例是 MyBean 的字段，用于以下示例代码。

```

@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();
    }
}

```

以下代码示例创建一个 InnerBean 对象，并仅为其两个属性中的一个设置了值。

```

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);
}

```

为了可视化发送到 DynamoDB 的低级别数据，该代码会在保存 MyBean 对象之前记录属性映射。

记录的输出显示，innerBeanWithIgnoreNullsAnno 输出了一个属性，

```
innerBeanWithIgnoreNullsAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200)}))
```

`innerBeanWithoutAnno` 实例输出了两个属性。一个属性的值为 200，另一个属性的值为空。

```
innerBeanWithoutAnno=AttributeValue(M={innerBeanFieldInteger=AttributeValue(N=200),  
innerBeanFieldString=AttributeValue(NUL=true)})
```

属性映射的 JSON 表示

以下 JSON 表示可以更轻松地查看保存到 DynamoDB 的数据。

```
{  
  "id": {  
    "S": "1"  
  },  
  "innerBeanWithIgnoreNullsAnno": {  
    "M": {  
      "innerBeanFieldInteger": {  
        "N": "200"  
      }  
    }  
  },  
  "innerBeanWithoutAnno": {  
    "M": {  
      "innerBeanFieldInteger": {  
        "N": "200"  
      },  
      "innerBeanFieldString": {  
        "NULL": true  
      }  
    }  
  }  
}
```

替代静态架构

您可以使用以下 `StaticTableSchema` 版本的表架构来代替数据类标注。

```
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();
}
```

使用适用于 DynamoDB 的增强型文档 API 处理 JSON 文档

AWS SDK for Java 2.x 的[增强型文档 API](#)旨在处理没有固定架构的面向文档的数据。该 API 也允许您使用自定义类来映射单个属性。

增强型文档 API 是AWS SDK for Java v1.x [文档 API](#) 的后继版本。

目录

- [开始使用增强型文档 API](#)
- [创建 DocumentTableSchema 和 DynamoDbTable。](#)

- [生成增强型文档](#)
 - [使用 JSON 字符串生成](#)
 - [基于单个元素构建](#)
- [执行 CRUD 操作](#)
- [将增强型文档属性作为自定义对象进行访问](#)
- [在没有 DynamoDB 的情况下使用 EnhancedDocument](#)

开始使用增强型文档 API

增强型文档 API 所需的[依赖项](#)与 DynamoDB 增强型客户端 API 所需的依赖项相同。它还需要一个[DynamoDbEnhancedClient 实例](#)，如本主题开头所示。

由于增强型文档 API 是在 AWS SDK for Java 2.x 2.20.3 版本中发布的，因此您需要使用该版本或更高版本。

创建 **DocumentTableSchema** 和 **DynamoDbTable**。

要使用增强型文档 API 对 DynamoDB 表调用命令，请将该表与客户端[DynamoDbTable<EnhancedDocument>](#) 资源对象关联。

增强型客户端的 `table()` 方法会创建一个 `DynamoDbTable<EnhancedDocument>` 实例，并且需要用于 DynamoDB 表名称和 `DocumentTableSchema` 的参数。

[DocumentTableSchema](#) 的生成器需要一个主索引键和一个或多个属性转换器提供程序。`AttributeConverterProvider.defaultProvider()` 方法为[默认类型](#)提供转换器。即使您提供了自定义属性转换器提供程序，也应指定它。您可以向生成器添加可选的二级索引键。

以下代码段显示的代码将生成一个 DynamoDB person 表的客户端表示形式，该表将存储无架构 `EnhancedDocument` 对象。

```
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.
```

以下显示了本部分中使用的 person 对象的 JSON 表示形式。

JSON person 对象

```
{
  "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"
    }
  ]
}
```

```
]
}
```

生成增强型文档

[EnhancedDocument](#) 表示具有复杂结构和嵌套属性的文档类型对象。EnhancedDocument 需要与为 DocumentTableSchema 指定的主键属性相匹配的顶级属性。其余内容是任意的，可以由顶级属性以及深度嵌套的属性组成。

您可以使用提供多种元素添加方法的生成器来创建 EnhancedDocument 实例。

使用 JSON 字符串生成

使用 JSON 字符串，您可以在一个方法调用中生成 EnhancedDocument。以下代码段从 jsonPerson() 帮助程序方法返回的 JSON 字符串创建一个 EnhancedDocument。jsonPerson() 方法返回前面显示的 [person 对象](#) 的 JSON 字符串版本。

```
EnhancedDocument document =
    EnhancedDocument.builder()
        .json( jsonPerson() )
        .build();
```

基于单个元素构建

或者，您可以使用生成器的类型安全方法从各个组件生成 EnhancedDocument 实例。

以下示例生成一个 person 增强型文档，该文档类似于上一个示例中从 JSON 字符串生成的增强型文档。

```
/* 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();

```

帮助程序方法

```

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"));
}
```

执行 CRUD 操作

定义 `EnhancedDocument` 实例后，您可以将其保存到 DynamoDB 表。以下代码段使用基于单个元素创建的 [personDocument](#)。

```
documentDynamoDbTable.putItem(personDocument);
```

读取 DynamoDB 中的增强型文档实例后，您可以使用 `getter` 提取各个属性值，如以下代码段所示，这些代码段用于访问从 `personDocument` 中保存的数据。或者，您可以将完整内容提取为 JSON 字符串，如示例代码的最后一部分所示。

```
// 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 实例可以与 [DynamoDbTable](#) 的任何方法或 [DynamoDbEnhancedClient](#) 结合使用来代替映射的数据类。

将增强型文档属性作为自定义对象进行访问

除了提供用于读取和写入具有无架构结构的属性的 API 之外，增强型文档 API 还允许您在自定义类的实例之间转换属性。

增强型文档 API 使用[控制属性转换](#)部分中显示的 AttributeConverterProvider 和 AttributeConverter，作为 DynamoDB 增强型客户端 API 的一部分。

在以下示例中，我们使用 `CustomAttributeConverterProvider` 及其嵌套 `AddressConverter` 类来转换 `Address` 对象。

此示例表明，您可以混合类中的数据以及根据需要构建的结构中的数据。此示例还表明，自定义类可以在嵌套结构的任何级别上使用。此示例中的 `Address` 对象是映射中使用的值。

```
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 getAddressessForCustomMappingExample() 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 = getAddressessForCustomMappingExample();

    // 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 代码

```

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 object.
    @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 类

```
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;
}
}
```

提供地址的帮助程序方法

以下帮助程序方法提供的映射使用自定义 `Address` 实例作为值，而不是使用通用 `Map<String, String>` 实例作为值。

```
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);
}
```

在没有 DynamoDB 的情况下使用 **EnhancedDocument**

尽管您通常使用 `EnhancedDocument` 的实例来读取和写入文档类型的 DynamoDB 项目，但它也可以独立于 DynamoDB 使用。

您可以使用 `EnhancedDocuments` 的功能，在 JSON 字符串或自定义对象之间，执行到 `AttributeValues` 的低级映射的转换，如以下示例所示。

```
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

当您独立于 DynamoDB 表使用增强型文档时，请务必在生成器上明确设置属性转换器提供程序。

相比之下，当增强型文档与 DynamoDB 表结合使用时，文档表架构会提供转换器提供程序。

使用扩展

DynamoDB 增强型客户端 API 支持插件扩展，这些插件扩展提供的功能不仅限于映射操作。扩展有两种钩子方法，`beforeWrite()` 和 `afterRead()`。`beforeWrite()` 在写入操作发生之前对其进行修改，该 `afterRead()` 方法在读取操作发生后修改其结果。由于某些操作（例如项目更新）同时执行写入和读取，因此两种钩子方法都会被调用。

扩展按增强型客户端生成器中指定的顺序加载。加载顺序可能很重要，因为一个扩展可以对前一个扩展变换过的值起作用。

增强型客户端 API 附带了一组插件扩展，位于 [extensions](#) 包中。默认情况下，增强型客户端会加载 [VersionedRecordExtension](#) 和 [AtomicCounterExtension](#)。您可以使用增强型客户端生成器覆盖默认行为并加载任何扩展。如果您不想使用默认扩展，也可以指定一个都不用。

如果您加载自己的扩展，则增强型客户端不会加载任何默认扩展。如果您想要任一默认扩展提供的行为，则需要将其明确添加到扩展列表中。

在以下示例中，名为 `verifyChecksumExtension` 的自定义扩展是在 `VersionedRecordExtension` 之后加载的，该扩展通常在默认情况下自行加载。本示例中未加载 `AtomicCounterExtension`。

```
DynamoDbEnhancedClientExtension versionedRecordExtension =
    VersionedRecordExtension.builder().build();

DynamoDbEnhancedClient enhancedClient =
    DynamoDbEnhancedClient.builder()
        .dynamoDbClient(dynamoDbClient)
        .extensions(versionedRecordExtension,
            verifyChecksumExtension)
        .build();
```

VersionedRecordExtension

默认情况下会加载 `VersionedRecordExtension`，当项目写入数据库时，它会递增和跟踪项目的版本号。如果实际永久保存的项目的版本号与应用程序上次读取的值不匹配，则会在每次写入时添加一个导致写入失败的条件。此行为有效地为项目更新提供了乐观锁。如果另一个进程在第一个进程读取项目到写入更新内容之间更新该项目，则写入操作将失败。

要指定使用哪个属性来跟踪项目版本号，请在表架构中标记数字属性。

以下代码段指定 `version` 属性应包含项目版本号。

```
@DynamoDbVersionAttribute
public Integer getVersion() {...};
public void setVersion(Integer version) {...};
```

下面的代码段显示了等效的静态表架构方法。

```
.addAttribute(Integer.class, a -> a.name("version")
                .getter(Customer::getVersion)
                .setter(Customer::setVersion)
                // Apply the 'version' tag to the attribute.

.tags(VersionedRecordExtension.AttributeTags.versionAttribute())
```

AtomicCounterExtension

默认情况下会加载 `AtomicCounterExtension`，并且每次向数据库写入记录时都会增加一个带标签的数字属性。可以指定起始值和增量值。如果未指定任何值，则将起始值设置为 0，属性的值以 1 为增量。

要指定哪个属性是计数器，请在表架构中标记一个类型为 `Long` 的属性。

以下代码段显示了为 `counter` 属性使用默认起始值和增量值的情况。

```
@DynamoDbAtomicCounter
public Long getCounter() {...};
public void setCounter(Long counter) {...};
```

下面的代码段显示了静态表架构方法。原子计数器扩展使用起始值 10，并在每次写入记录时将该值递增 5。

```
.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

每次成功将项目写入数据库时，`AutoGeneratedTimestampRecordExtension` 都会自动使用当前时间戳更新类型为 `Instant` 的已标记属性。

默认情况下不加载此扩展。因此，在构建增强型客户端时，您需要将其指定为自定义扩展，如本主题的第一个示例所示。

要指定将使用当前时间戳更新的属性，请在表架构中标记 `Instant` 属性。

在以下代码段中，`lastUpdate` 属性是扩展行为的目标。请注意该属性必须是 `Instant` 类型的要求。

```
@DynamoDbAutoGeneratedTimestampAttribute
public Instant getLastUpdate() {...}
public void setLastUpdate(Instant lastUpdate) {...}
```

下面的代码段显示了等效的静态表架构方法。

```
.addAttribute(Instant.class, a -> a.name("lastUpdate")
    .getter(Customer::getLastUpdate)
    .setter(Customer::setLastUpdate)
    // Applying the 'autoGeneratedTimestamp' tag to
the attribute.

.tags(AutoGeneratedTimestampRecordExtension.AttributeTags.autoGeneratedTimestampAttribute())
```

自定义扩展

以下自定义扩展类显示了使用更新表达式的 `beforeWrite()` 方法。在注释行 2 之后，如果数据库中的项目还没有 `registrationDate` 属性，我们会创建一个 `SetAction` 来设置 `registrationDate` 属性。每当更新 `Customer` 对象时，扩展都会确保设置了 `registrationDate`。

```
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();
    }
}

```

异步使用 DynamoDB 增强型客户端 API

如果您的应用程序需要对 DynamoDB 进行非阻塞异步调用，则可以使用 [DynamoDbEnhancedAsyncClient](#)。它与同步实现类似，但有以下主要区别：

1. 构建时 `DynamoDbEnhancedAsyncClient`，必须提供标准客户端的异步版本 `DynamoDbAsyncClient`，如以下代码段所示。

```

DynamoDbEnhancedAsyncClient enhancedClient =
    DynamoDbEnhancedAsyncClient.builder()
        .dynamoDbAsyncClient(dynamoDbAsyncClient)
        .build();

```

2. 返回单个数据对象的方法会返回结果的 `CompletableFuture`，而不仅仅是结果。然后，您的应用程序可以执行其他工作，而不必因结果阻塞。以下代码段显示了异步 `getItem()` 方法。

```
CompletableFuture<Customer> result = customerDynamoDbTable.getItem(customer);
// Perform other work here.
return result.join(); // Now block and wait for the result.
```

3. 返回分页结果列表的方法会返回 [SdkIterable](#)，而不是同步 `DynamoDbEnhanceClient` 会为相同方法返回的 [SdkPublisher](#)。然后，您的应用程序可以向该发布者订阅处理程序，以异步方式处理结果，而无需阻塞。

```
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.
```

有关更完整的 `SdkPublisher` API 使用示例，请参阅本指南中讨论异步 `scan()` 方法的部分中的 [示例](#)。

数据类注释

下表列出了可用于数据类的注释，并提供了指向本指南中信息和示例的链接。该表按注释名称的字母升序排序。

本指南中使用的数据类注释

注释名称	注释适用于 ¹	作用	本指南中显示的位置
<code>DynamoDbAtomicCounter</code>	属性 ²	每次向数据库写入记录时都会增加一个带标签的数字属性。	介绍和讨论。
<code>DynamoDbAttribute</code>	属性	定义或重命名映射到 DynamoDB 表属性的 Bean 属性。	<ul style="list-style-type: none"> 初步讨论。 入门部分，请参阅“注意”。 MovieActor 课堂上的 In Query 方法示例。

注释名称	注释适用于 ¹	作用	本指南中显示的位置
DynamoDbAutoGeneratedTimestampAttribute	属性	每次成功将项目写入数据库时，都使用当前时间戳更新已标记的属性	介绍和讨论。
DynamoDbBean	class	将数据类标记为可映射到表架构。	第一次是在“入门”部分的 Customer 类 上使用。本指南中展示了几种用法。
DynamoDbConvertedBy	属性	将自定义 Attribute Converter 与带注释的属性相关联。	初步讨论和示例。
DynamoDbFlatten	属性	扁平化单独的 DynamoDB 数据类的所有属性，并将它们作为顶级属性添加到从数据库读取的记录和写入数据库的记录中。	<ul style="list-style-type: none"> • 初步讨论。 • 对其他代码的影响。
DynamoDbIgnore	属性	导致属性保持未映射状态。	<ul style="list-style-type: none"> • 初步讨论。 • 在 ProductCatalog 课堂上使用。
DynamoDbIgnoreNulls	属性	防止保存嵌套 DynamoDb 对象的空属性。	讨论和示例。
DynamoDbImmutable	class	将不可变数据类标记为可映射到表架构。	<ul style="list-style-type: none"> • 注释简介。 • 在 ProductCatalog 课堂上使用。 • 与 Lombok 结合使用。

注释名称	注释适用于 ¹	作用	本指南中显示的位置
DynamoDbPartitionKey	属性	将属性标记为 DynamoDb 表的主分区键（哈希键）。	<ul style="list-style-type: none"> • 第一次是在“入门”部分的 Customer 类上使用。 • 与 Lombok 结合使用。
DynamoDbP reserveEmptyObject	属性	如果映射到带注释的属性的对象没有数据，则指定应使用所有空字段初始化该对象。	讨论和示例。
DynamoDbS econdaryPartitionKey	属性	将属性标记为全局二级属性的分区键。	<ul style="list-style-type: none"> • 在二级索引和示例中使用。 • 在查询方法示例中。 • 在 Lombok 示例中。 • 与不可变类结合使用。
DynamoDbS econdarySortKey	属性	将属性标记为全局或本地二级索引的可选排序键。	<ul style="list-style-type: none"> • 在二级索引和示例中使用。 • 在查询方法示例中。 • 在 Lombok 示例中。 • 与不可变类结合使用。

注释名称	注释适用于 ¹	作用	本指南中显示的位置
DynamoDbSortKey	属性	将属性标记为可选的主排序键（范围键）。	<ul style="list-style-type: none"> “入门”部分的 Customer 类上。 与不可变类结合使用。 在 Lombok 示例中。 在查询方法示例中。
DynamoDbUpdateBehavior	属性	指定作为“更新”操作的一部分更新此属性时的行为，例如 UpdateItem。	简介和示例 。
DynamoDbVersionAttribute	属性	递增项目版本号。	介绍和讨论 。

¹ 你可以将属性级注释应用于 getter 或 setter，但不能同时应用两者。本指南显示了 getter 上的注释。

² 该术语 property 通常用于封装在数据类中的值。JavaBean 但是，为了与 DynamoDB 使用的术语保持一致，本指南（英文版）改用术语 attribute。（中文版中，两者的翻译均为“属性”。）

与... 一起工作 Amazon EC2

本节提供使用 AWS SDK for Java 2.x [Amazon EC2](#) 的编程示例。

主题

- [托管 Amazon EC2 实例](#)
- [使用 AWS 区域、区和可用区](#)
- [在中使用安全组 Amazon EC2](#)
- [使用 Amazon EC2 实例元数据](#)

托管 Amazon EC2 实例

创建实例

要创建新 Amazon EC2 实例，请调用 [Ec2Client](#) 的 [runInstances](#) 方法，并为它提供 [RunInstancesRequest](#)，其中包含要使用的[亚马逊机器映像 \(AMI\)](#) 和一个[实例类型](#)。

导入

```
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;
```

代码

```
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 "";
}
```

请参阅 GitHub 上的[完整示例](#)。

启动实例

要启动 Amazon EC2 实例，请调用 `Ec2Client` 的 [startInstances](#) 方法，并为它提供 [StartInstancesRequest](#)，其中包含要启动实例的 ID。

导入

```
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;
```

代码

```
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);
}
```

请参阅 GitHub 上的[完整示例](#)。

停止实例

要停止 Amazon EC2 实例，请调用 `Ec2Client` 的 [stopInstances](#) 方法，并为它提供 [StopInstancesRequest](#)，其中包含要停止的实例的 ID。

导入

```
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;
```

代码

```
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);
}
```

请参阅 GitHub 上的[完整示例](#)。

重启实例

要重启 Amazon EC2 实例，请调用 `Ec2Client` 的 [rebootInstances](#) 方法，并为它提供 [RebootInstancesRequest](#)，其中包含要重启实例的 ID。

导入

```
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;
```

代码

```
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);
}
}
```

请参阅 GitHub 上的[完整示例](#)。

描述实例

要列出您的实例，您需要创建 [DescribeInstancesRequest](#) 并调用 Ec2Client 的 [describeInstances](#) 方法。该方法将返回 [DescribeInstancesResponse](#) 对象，您可以用它来列出您的账户和区域的 Amazon EC2 实例。

实例按预留进行分组。每个预留对应启动实例的 `startInstances` 的调用。要列出您的实例，您必须先调用 `DescribeInstancesResponse` 类的 `reservations` 方法，然后在每个返回的 `instancesReservation` [对象上调用](#)。

导入

```
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;
```

代码

```
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);  
}  
}
```

结果将分页；您可以获取更多结果，方法是将从结果对象的 `nextToken` 方法返回的值传递到新请求对象的 `nextToken` 方法，然后在下一个 `describeInstances` 调用中使用新请求对象。

请参阅 GitHub 上的[完整示例](#)。

监控实例

您可以监控 Amazon EC2 实例的各方面，例如 CPU 和网络利用率、可用内存和剩余磁盘空间。要了解有关实例监控的信息，请参阅《Amazon EC2 用户指南（适用于 Linux 实例）》中的[监控 Amazon EC2](#)。

要开始监控实例，您必须用要监控实例的 ID 创建一个 [MonitorInstancesRequest](#)，并将其传递给 `Ec2Client` 的 [monitorInstances](#) 方法。

导入

```
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;
```

代码

```
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);
}
```

请参阅 GitHub 上的[完整示例](#)。

停止实例监控

要停止监控实例，您必须用要停止监控实例的 ID 创建一个 [UnmonitorInstancesRequest](#)，并将其传递给 Ec2Client 的 [unmonitorInstances](#) 方法。

导入

```
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;
```

代码

```
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);
}
```

请参阅 GitHub 上的[完整示例](#)。

更多信息

- 《Amazon EC2 API Reference》中的 [RunInstances](#)
- 《Amazon EC2 API Reference》中的 [DescribeInstances](#)
- 《Amazon EC2 API Reference》中的 [StartInstances](#)
- 《Amazon EC2 API Reference》中的 [StopInstances](#)
- 《Amazon EC2 API Reference》中的 [RebootInstances](#)
- 《Amazon EC2 API Reference》中的 [MonitorInstances](#)
- 《Amazon EC2 API Reference》中的 [UnmonitorInstances](#)

使用 AWS 区域 区和可用区

描述区域

要列出账户可用的区域，请调用 `Ec2Client` 的 `describeRegions` 方法。它返回 [DescribeRegionsResponse](#)。调用返回对象的 `regions` 方法，获取表示各个区域的 [Region](#) 对象的列表。

导入

```
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

代码

```
try {
    DescribeRegionsResponse regionsResponse = ec2.describeRegions();
    regionsResponse.regions().forEach(region -> {
```

```
        System.out.printf(
            "Found Region %s with endpoint %s%n",
            region.regionName(),
            region.endpoint());
        System.out.println();
    });
```

请参阅上的[完整示例](#) GitHub。

描述可用区

要列出账户可用的每个可用区，请调用 `Ec2Client` 的 `describeAvailabilityZones` 方法。它返回 [DescribeAvailabilityZonesResponse](#)。调用其 `availabilityZones` 方法以获取代表每个可用区的 [AvailabilityZone](#) 对象列表。

导入

```
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

代码

创建 `Ec2Client`。

```
software.amazon.awssdk.regions.Region region =
software.amazon.awssdk.regions.Region.US_EAST_1;
Ec2Client ec2 = Ec2Client.builder()
    .region(region)
    .build();
```

然后调用 `describeAvailabilityZones ()` 并检索结果。

```
DescribeAvailabilityZonesResponse zonesResponse =
ec2.describeAvailabilityZones();
zonesResponse.availabilityZones().forEach(zone -> {
    System.out.printf(
        "Found Availability Zone %s with status %s in region %s%n",
        zone.zoneName(),
```



```
        zone.state(),
        zone.regionName()
    );
    System.out.println();
});
```

请参阅上的[完整示例](#) GitHub。

描述账户

要列出有关账户的 EC2 相关信息，请调用 `Ec2Client` 的 `describeAccountAttributes` 方法。此方法返回一个 [DescribeAccountAttributesResponse](#) 对象。调用此对象 `accountAttributes` 方法以获取 [AccountAttribute](#) 对象列表。您可以遍历列表以检索 `AccountAttribute` 对象。

您可以通过调用 `AccountAttribute` 对象的 `attributeValues` 方法来获取账户的属性值。此方法返回 [AccountAttributeValue](#) 对象列表。您可以遍历第二个列表来显示属性的值（请参阅以下代码示例）。

导入

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeAccountAttributesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

代码

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeAccountAttributesResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAccount {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        Ec2Client ec2 = Ec2Client.builder()
```

```
        .region(region)
        .build();

    describeEC2Account(ec2);
    System.out.print("Done");
    ec2.close();
}

public static void describeEC2Account(Ec2Client ec2) {
    try {
        DescribeAccountAttributesResponse accountResults =
ec2.describeAccountAttributes();
        accountResults.accountAttributes().forEach(attribute -> {
            System.out.print("\n The name of the attribute is " +
attribute.attributeName());
            attribute.attributeValues().forEach(
                myValue -> System.out.print("\n The value of the attribute is "
+ myValue.attributeValue()));
        });

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

请参阅上的[完整示例](#) GitHub。

更多信息

- Linux 实例 Amazon EC2 用户指南中的@@ [区域和可用区](#)
- [DescribeRegions](#)在 Amazon EC2 API 参考中
- [DescribeAvailabilityZones](#)在 Amazon EC2 API 参考中

在中使用安全组 Amazon EC2

创建安全组

要创建安全组，请使用包含密钥名称的 Ec2Client createSecurityGroup 方法调用。[CreateSecurityGroupRequest](#)

导入

```
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;
```

代码

```
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
        .groupName(groupName)
        .description(groupDesc)
        .vpcId(vpcId)
        .build();

        CreateSecurityGroupResponse resp= ec2.createSecurityGroup(createRequest);
```

请参阅上的[完整示例](#) GitHub。

配置安全组

安全组可以控制您的 Amazon EC2 实例的入站 (入口) 和出站 (出口) 流量。

要向您的安全组添加入口规则，请使用 `Ec2Client authorizeSecurityGroupIngress` 的方法，在对象中提供安全组的名称和要分配给它的访问规则

([IpPermission](#))。 [AuthorizeSecurityGroupIngressRequest](#)以下示例演示如何将 IP 权限添加到安全组。

导入

```
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;
```

代码

首先，创建一个 Ec2Client

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

然后使用 Ec2Client 的 authorizeSecurityGroupIngress 方法，

```
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 "";
}
```

要向安全组添加出口规则，请在 `Ec2Client` 的 [AuthorizeSecurityGroupEgressRequest](#) 方法中提供类似的数据。 `authorizeSecurityGroupEgress`

请参阅上的 [完整示例](#) GitHub。

描述安全组

要描述您的安全组或获取相关信息，请调用 `Ec2Client` 的 `describeSecurityGroups` 方法。它返回一个 [DescribeSecurityGroupsResponse](#)，您可以使用该方法通过调用其 `securityGroups` 方法来访问安全组列表，该方法返回 [SecurityGroup](#) 对象列表。

导入

```
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;
```

代码

```
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);
}
}
```

请参阅上的[完整示例](#) GitHub。

删除安全组

要删除安全组，请调用 `Ec2Client deleteSecurityGroup` 的方法，将其传递给[DeleteSecurityGroupRequest](#)包含要删除的安全组 ID。

导入

```
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;
```

代码

```
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);  
    }  
}
```

请参阅上的[完整示例](#) GitHub。

更多信息

- Amazon EC2 Linux 实例 Amazon EC2 用户指南中的@@ [安全组](#)
- 在 [Linux 实例 Amazon EC2 用户指南中授权您的 Linux 实例的入站流量](#)
- [CreateSecurityGroup](#)在 Amazon EC2 API 参考中
- [DescribeSecurityGroups](#)在 Amazon EC2 API 参考中
- [DeleteSecurityGroup](#)在 Amazon EC2 API 参考中
- [AuthorizeSecurityGroupIngress](#)在 Amazon EC2 API 参考中

使用 Amazon EC2 实例元数据

适用于 Amazon EC2 实例元数据服务的 Java SDK 客户端 (元数据客户端) 允许您的应用程序访问其本地 EC2 实例上的元数据。元数据客户端使用 [IMDSv2](#) (实例元数据服务 v2) 的本地实例，并使用面向会话的请求。

SDK 中有两个客户端类可用。同步 [Ec2MetadataClient](#) 用于阻塞操作，[Ec2MetadataAsyncClient](#) 用于异步、非阻塞用例。

开始使用

要使用元数据客户端，请将 imds Maven 构件添加到您的项目中。您还需要在类路径上具有 [SdkHttpClient](#) 的类，或对于异步变体而言，具有 [SdkAsyncHttpClient](#) 的类。

以下 Maven XML 显示了使用同步 [URLConnectionHttpClient](#) 的依赖项片段以及元数据客户端的依赖项。

```
<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>

```

在 [Maven Central 存储库](#) 中搜索 bom 构件的最新版本。

要使用异步 HTTP 客户端，请替换 url-connection-client 构件的依赖项片段。例如，以下片段引入了 [NettyNioAsyncHttpClient](#) 实现。

```

<dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>netty-nio-client</artifactId>
</dependency>

```

使用元数据客户端

实例化元数据客户端

当类路径上只有一个 SdkHttpClient 接口的实现时，您可以实例化同步 Ec2MetadataClient 的实例。为此，请调用 static Ec2MetadataClient#create() 方法，如以下代码段所示。

```

Ec2MetadataClient client = Ec2MetadataClient.create(); //
'Ec2MetadataAsyncClient#create' is the asynchronous version.

```

如果您的应用程序有多个 SdkHttpClient 或 SdkHttpAsyncClient 接口的实现，则必须指定一个实现以供元数据客户端使用，如 [the section called “可配置 HTTP 客户端”](#) 部分所示。

Note

对于大多数服务客户端（例如 Amazon S3），适用于 Java 的 SDK 会自动添加 `SdkHttpClient` 或 `SdkHttpAsyncClient` 接口的实现。如果您的元数据客户端使用相同的实现，则 `Ec2MetadataClient#create()` 将起作用。如果您需要不同的实现，则必须在创建元数据客户端时指定它。

发送请求

要检索实例元数据，请实例化 `EC2MetadataClient` 类，然后使用指定[实例元数据类别](#)的路径参数调用 `get` 方法。

以下示例将与 `ami-id` 键关联的值打印到控制台。

```
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.
```

如果路径无效，则 `get` 方法将引发异常。

请对多个请求重复使用同一个客户端实例，但当不再需要该客户端时，请在该客户端上调用 `close` 来释放资源。调用 `close` 方法后，将无法再使用客户端实例。

解析响应

EC2 实例元数据可以以不同的格式输出。纯文本和 JSON 是最常用的格式。元数据客户端提供了使用这些格式的方法。

如以下示例所示，使用 `asString` 方法以 Java 字符串的形式获取数据。您还可以使用 `asList` 方法来分隔返回多行的纯文本响应。

```
Ec2MetadataClient client = Ec2MetadataClient.create();
Ec2MetadataResponse response = client.get("/latest/meta-data/");
String fullResponse = response.asString();
List<String> splits = response.asList();
```

如果响应采用 JSON 格式，请使用 `Ec2MetadataResponse#asDocument` 方法将 JSON 响应解析为 [Document](#) 实例，如以下代码段所示。

```
Document fullResponse = response.asDocument();
```

如果元数据的格式不是 JSON，则会引发异常。如果成功解析了响应，则可以使用 [document API](#) 来更详细地检查响应。请查阅实例[元数据类别表](#)，了解哪些元数据类别提供了 JSON 格式的响应。

配置元数据客户端

重试

您可以为元数据客户端配置重试机制。如果这样做，则客户端可以自动重试因意外原因而失败的请求。默认情况下，客户端对失败的请求重试三次，两次尝试之间的时间呈指数回退。

如果您的用例需要不同的重试机制，则可以使用其客户端生成器上的 `retryPolicy` 方法自定义客户端。例如，以下示例将同步客户端配置为共五次重试，每两次重试之间固定延迟两秒钟。

```
BackoffStrategy fixedBackoffStrategy =
    FixedDelayBackoffStrategy.create(Duration.ofSeconds(2));
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
        .retryPolicy(retryPolicyBuilder ->
            retryPolicyBuilder.numRetries(5)

            .backoffStrategy(fixedBackoffStrategy))
        .build();
```

您可以将多种 [BackoffStrategies](#) 与元数据客户端结合使用。

您也可以完全禁用重试机制，如以下代码段所示。

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
        .retryPolicy(Ec2MetadataRetryPolicy.none())
        .build();
```

使用 `Ec2MetadataRetryPolicy#none()` 会禁用默认的重试策略，因此元数据客户端不尝试重试。

IP 版本

默认情况下，元数据客户端使用位于 `http://169.254.169.254` 的 IPV4 端点。要将客户端更改为使用 IPV6 版本，请使用生成器的 `endpointMode` 或 `endpoint` 方法。如果在生成器上同时调用这两个方法，则会出现异常。

以下示例演示两个 IPV6 选项。

```
Ec2MetadataClient client =  
    Ec2MetadataClient.builder()  
        .endpointMode(EndpointMode.IPV6)  
        .build();
```

```
Ec2MetadataClient client =  
    Ec2MetadataClient.builder()  
        .endpoint(URI.create("http://[fd00:ec2::254]"))  
        .build();
```

主要特征

异步客户端

要使用非阻塞版本的客户端，请实例化 `Ec2MetadataAsyncClient` 类的实例。以下示例中的代码使用默认设置创建异步客户端，并使用 `get` 方法检索 `ami-id` 键的值。

```
Ec2MetadataAsyncClient asyncClient = Ec2MetadataAsyncClient.create();  
CompletableFuture<Ec2MetadataResponse> response = asyncClient.get("/latest/meta-data/  
ami-id");
```

当响应返回时，`get` 方法返回的 `java.util.concurrent.CompletableFuture` 就完成了。以下示例将 `ami-id` 元数据打印到控制台。

```
response.thenAccept(metadata -> System.out.println(metadata.asString()));
```

可配置 HTTP 客户端

每个元数据客户端的生成器都有一种可用于提供自定义 HTTP 客户端的 `httpClient` 方法。

以下示例显示了自定义 `URLConnectionHttpClient` 实例的代码。

```
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.
```

以下示例显示了带有异步元数据客户端的自定义 `NettyNioAsyncHttpClient` 实例的代码。

```
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.
```

本指南中的 [the section called “HTTP 客户”](#) 主题详细介绍了如何配置适用于 Java 的 SDK 中可用的 HTTP 客户端。

令牌缓存

由于元数据客户端使用 IMDSv2，因此所有请求都与会话关联。会话由带过期时间的令牌定义，元数据客户端会为您管理该令牌。每个元数据请求都会自动重复使用令牌，直到令牌过期。

默认情况下，令牌持续六小时（21600 秒）。除非您的特定用例需要高级配置，否则我们建议您保留默认的生存时间值。

如果需要，可使用 `tokenTtl` 生成器方法配置持续时间。例如，以下代码段中的代码创建了一个会话持续时间为五分钟的客户。

```
Ec2MetadataClient client =
    Ec2MetadataClient.builder()
        .tokenTtl(Duration.ofMinutes(5))
        .build();
```

如果您省略调用生成器上的 `tokenTtl` 方法，则改用默认持续时间 21,600。

使用 IAM

本部分提供使用 AWS SDK for Java 2.x 对 AWS Identity and Access Management (IAM) 进行编程的示例。

AWS Identity and Access Management (IAM) 使您能够安全地控制您的用户对 AWS 服务和资源的访问权限。使用 IAM，您可以创建和管理 AWS 用户和组，并使用权限来允许和拒绝他们对 AWS 资源的访问。有关 IAM 的完整说明，请访问《[IAM 用户指南](#)》。

以下示例仅包含演示每种方法所需的代码。[完整的示例代码在 GitHub 上提供](#)。您可以从中下载单个源文件，也可以将存储库复制到本地以获得所有示例，然后构建并运行它们。

主题

- [管理 IAM 访问密钥](#)
- [管理 IAM 用户](#)
- [使用创建 IAM 策略 AWS SDK for Java 2.x](#)
- [使用 IAM 策略](#)
- [使用 IAM 服务器证书](#)

管理 IAM 访问密钥

创建访问密钥

要创建 IAM 访问密钥，请使用 [CreateAccessKeyRequest](#) 对象调用该 `IamClient` 的 `createAccessKey` 方法。

Note

您必须将区域设置为 `AWS_GLOBAL` 才能使 `IamClient` 呼叫生效，因为 IAM 这是一项全球服务。

导入

```
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;
```

代码

```
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 "";
}
```

请参阅上的[完整示例](#) GitHub。

列出访问密钥

要列出给定用户的访问密钥，请创建一个包含要列出密钥的用户名的[ListAccessKeysRequest](#)对象，然后将其传递给IamClient'slistAccessKeys方法。

Note

如果您不向提供用户名listAccessKeys，它将尝试列出与签署请求的用户关联 AWS 账户的访问密钥。

导入

```
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;
```

代码

```
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);
    }
}
```

`listAccessKeys` 的结果分页显示 (默认情况下, 每个调用最多返回 100 个记录)。您可以调用返回 `isTruncated` 的 [ListAccessKeysResponse](#) 对象, 以查看查询返回的结果是否少于可用的结果。如果是, 则调用 `marker` 中的 `ListAccessKeysResponse` 并在创建新请求时使用它。在下次调用 `listAccessKeys` 时使用该新请求。

请参阅上的 [完整示例](#) GitHub。

检索上次使用访问密钥的时间

要获取上次使用访问密钥的时间, 请使用访问密钥的 ID 调用该 `IamClient` 的 `getAccessKeyLastUsed` 方法 (可以使用 [GetAccessKeyLastUsedRequest](#) 对象传入)。

然后, 您可以使用返回的 [GetAccessKeyLastUsedResponse](#) 对象来检索密钥的上次使用时间。

导入

```
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;
```

代码

```
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");
}
```



```
}
```

请参阅上的[完整示例](#) GitHub。

激活或停用访问密钥

您可以激活或停用访问密钥，方法是创建[UpdateAccessKeyRequest](#)对象，提供访问密钥 ID、可选的用户名和所需的访问密钥 [status](#)，然后将请求对象传递给 `IamClient` 的 `updateAccessKey` 方法。

导入

```
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;
```

代码

```
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);
    }
}
```

请参阅上的[完整示例](#) GitHub。

删除访问密钥

要永久删除访问密钥，请调用 `IamClient` 的 `deleteKey` 方法，为其提供 [DeleteAccessKeyRequest](#) 包含访问密钥的 ID 和用户名的访问密钥。

Note

密钥在删除后无法再检索或使用。要暂时停用密钥以便稍后可以再次激活，请改用 [updateAccessKey](#) 方法。

导入

```
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;
```

代码

```
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);  
    }  
}
```

请参阅上的[完整示例](#) GitHub。

更多信息

- [CreateAccessKey](#)在 IAM API 参考中
- [ListAccessKeys](#)在 IAM API 参考中
- [GetAccessKeyLastUsed](#)在 IAM API 参考中
- [UpdateAccessKey](#)在 IAM API 参考中
- [DeleteAccessKey](#)在 IAM API 参考中

管理 IAM 用户

创建用户

通过使用包含IAM用户名的[CreateUserRequest](#)对象向 iamClient的createUser方法提供用户名来创建新用户。

导入

```
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;
```

代码

```
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 "";
```

请参阅上的[完整示例](#) GitHub。

列出 用户

要列出您账户的IAM用户，请创建一个新用户 [ListUsersRequest](#) 并将其传递给 `IamClient` 的 `listUsers` 方法中。您可以通过调用 `users` 返回的 [ListUsersResponse](#) 对象来检索用户列表。

`listUsers` 返回的用户列表已分页。您可以通过调用响应对象的 `isTruncated` 方法查看更多可检索的结果。如果它返回 `true`，则调用响应对象的 `marker()` 方法。使用标记值创建新的请求对象。然后使用新请求再次调用 `listUsers` 方法。

导入

```
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;
```

代码

```
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);
    }
}
```

请参阅上的[完整示例](#) GitHub。

更新用户

要更新用户，请调用该 `IamClient` 对象 `updateUser` 的方法，该方法采用一个可用于更改用户名或路径的 [UpdateUserRequest](#) 对象。

导入

```
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;
```

代码

```
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);
    }
}
```

请参阅上的 [完整示例](#) GitHub。

删除用户

要删除用户，请使用设置为要删除 `IamClient` 的用户名的 [UpdateUserRequest](#) 对象调用 `deleteUser` 请求。

导入

```
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;
```

代码

```
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);
    }
}
```

请参阅上的[完整示例](#) GitHub。

更多信息

- 《IAM 用户指南》中的 [IAM 用户](#)
- 《IAM 用户指南》中的 [管理 IAM 用户](#)
- [CreateUser](#) 在 IAM API 参考中
- [ListUsers](#) 在 IAM API 参考中
- [UpdateUser](#) 在 IAM API 参考中
- [DeleteUser](#) 在 IAM API 参考中

使用创建 IAM 策略 AWS SDK for Java 2.x

[IAM 策略生成器 API](#) 是一个库，可用于在 Java 中构建 [IAM 策略](#) 并将其上传到 AWS Identity and Access Management (IAM)。

API 不是通过手动组装 JSON 字符串或读取文件来生成 IAM policy，而是提供了一种面向对象的客户端方法来生成 JSON 字符串。当您读取 JSON 格式的现有 IAM 策略时，API 会将其转换为 [IamPolicy](#) 实例进行处理。

IAM Policy 生成器 API 从 SDK 的 2.20.105 版本开始可用，因此请在 Maven 构建文件中使用该版本或更高版本。SDK 的最新版本号在 [Maven central 上列出](#)。

以下代码段显示了 Maven pom.xml 文件的依赖项代码块示例。该代码块允许您在项目中使用 IAM policy 生成器 API。

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>iam-policy-builder</artifactId>
  <version>2.20.139</version>
</dependency>
```

创建 IamPolicy

本部分显示了如何使用 IAM policy 生成器 API 生成策略的几个示例。

以下每个示例都从 [IamPolicy.Builder](#) 开始，然后使用 `addStatement` 方法添加一条或多条语句。按照这种模式，[IamStatement.Builder](#) 提供了向语句添加效果、操作、资源和条件的方法。

示例：创建基于时间的策略

以下示例创建了一个基于身份的策略，该策略允许在两个时间点之间执行 Amazon DynamoDB `GetItem` 操作。

```
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 输出

以上示例中的最后一条语句返回以下 JSON 字符串。

有关此[示例](#)的更多信息，请参阅《AWS Identity and Access Management 用户指南》。

```
{
  "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"
      }
    }
  }
}
```

示例：指定多个条件

以下示例演示如何创建基于身份的策略，以允许访问特定 DynamoDB 属性。该策略包含两个条件。

```
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 输出

以上示例中的最后一条语句返回以下 JSON 字符串。

有关此[示例](#)的更多信息，请参阅《AWS Identity and Access Management 用户指南》。

```

{
  "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"
      }
    }
  }
}

```

示例：指定主体

以下示例演示如何创建基于资源的策略，以拒绝除条件中指定的主体之外的所有主体访问某个桶。

```
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 输出

以上示例中的最后一条语句返回以下 JSON 字符串。

有关此[示例](#)的更多信息，请参阅《AWS Identity and Access Management 用户指南》。

```
{
  "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"
      }
    }
  }
}
```

示例：允许跨账户存取。

以下示例说明如何允许其他人将对象上传 AWS 账户 到您的存储桶，同时保留所有者对上传对象的完全控制权。

```
public String allowCrossAccountAccessExample() {
    IamPolicy policy = IamPolicy.builder()
        .addStatement(b -> b
            .effect(IamEffect.ALLOW)
            .addPrincipal(IamPrincipalType.AWS, "111122223333")
            .addAction("s3:PutObject")
            .addResource("arn:aws:s3:::DOC-EXAMPLE-BUCKET/*")
            .addCondition(b1 -> b1
                .operator(IamConditionOperator.STRING_EQUALS)
                .key("s3:x-amz-acl")
                .value("bucket-owner-full-control")))
        .build();
    return policy.toJson(IamPolicyWriter.builder()
        .prettyPrint(true).build());
}
```

JSON 输出

以上示例中的最后一条语句返回以下 JSON 字符串。

有关此[示例](#)的更多信息，请参阅《Amazon Simple Storage Service 用户指南》。

```
{
  "Version" : "2012-10-17",
  "Statement" : {
    "Effect" : "Allow",
    "Principal" : {
      "AWS" : "111122223333"
    },
    "Action" : "s3:PutObject",
    "Resource" : "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*",
    "Condition" : {
      "StringEquals" : {
        "s3:x-amz-acl" : "bucket-owner-full-control"
      }
    }
  }
}
```

将 **IamPolicy** 与 IAM 配合使用

创建 **IamPolicy** 实例后，您可以通过 [IamClient](#) 来使用 IAM 服务。

以下示例构建了一个策略，允许 [IAM 身份](#) 向使用 `accountID` 参数指定的账户中的 DynamoDB 表写入项目。然后，策略会作为 JSON 字符串上传到 IAM。

```
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());
}
```

下一个示例建立在前一个示例的基础上。该代码下载策略，并通过复制和修改声明将其用作新策略的基础。然后上传新策略。

```
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

前面的示例使用了一个 `IamClient` 参数，它是使用如下所示的代码段创建的。

```
IamClient iam = IamClient.builder().region(Region.AWS_GLOBAL).build();
```

JSON 格式的策略

这些示例返回以下 JSON 字符串。

```

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"
}
}
```

使用 IAM 策略

创建策略

要创建新策略，请在方法中提供策略名称和 JSON 格式 [CreatePolicyRequest](#) `IamClient` 的 `createPolicy` 策略文档。

导入

```
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;
```

代码

```
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 "";
}
```

请参阅上的[完整示例](#) GitHub。

获取策略

要检索现有策略，请调用 `IamClient`'s `getPolicy` 方法，在对象中提供策略的 ARN。 [GetPolicyRequest](#)

导入

```
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;
```

代码

```
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);
    }
}
```

请参阅上的[完整示例](#) GitHub。

附加角色策略

您可以通过调用 `IamClient`'s `attachRolePolicy` 方法将策略附加到IAM[角色](#)，并在中为其提供角色名称和策略 ARN。 [AttachRolePolicyRequest](#)

导入

```
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;
```

代码

```
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");
}
```

请参阅上的[完整示例](#) GitHub。

列出附加的角色策略

通过调用 `listAttachedRolePolicies` 方法列出角色 `IamClient` 的附加策略。它需要一个包含角色名称的 [ListAttachedRolePoliciesRequest](#) 对象来列出其策略。

调用返回 `getAttachedPolicies` 的 [ListAttachedRolePoliciesResponse](#) 对象以获取附加策略的列表。结果可能被截断；如果 `ListAttachedRolePoliciesResponse` 对象的 `isTruncated` 方法返回了 `true`，请调用 `ListAttachedRolePoliciesResponse` 对象的 `marker` 方法。使用返回的标记创建新请求并使用该请求再次调用 `listAttachedRolePolicies` 以获取下一批结果。

导入

```
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;
```

代码

```
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");
}
```

请参阅上的[完整示例](#) GitHub。

分离角色策略

要将策略与角色分离，请调用 `IamClient`'s `detachRolePolicy` 方法，在中为其提供角色名称和策略 ARN。 [DetachRolePolicyRequest](#)

导入

```
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;
```

代码

```
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);
    }
}
```

请参阅上的[完整示例](#) GitHub。

更多信息

- 《IAM 用户指南》中的 [IAM 策略概述](#)。
- 《IAM 用户指南》中的 [AWS IAM policy 参考](#)。
- [CreatePolicy](#)在 IAM API 参考中
- [GetPolicy](#)在 IAM API 参考中
- [AttachRolePolicy](#)在 IAM API 参考中
- [ListAttachedRolePolicies](#)在 IAM API 参考中
- [DetachRolePolicy](#)在 IAM API 参考中

使用 IAM 服务器证书

要启用与您的网站或应用程序的 HTTPS 连接 AWS，您需要一个 SSL/TLS 服务器证书。您可以使用外部提供商提供的 AWS Certificate Manager 服务器证书，也可以使用从外部提供商处获得的服务器证书。

我们建议您使用 ACM 来预置、管理和部署服务器证书。借助此功能，ACM 您可以申请证书，将其部署到您的 AWS 资源中，然后让我们为您 ACM 处理证书续订。提供的证书 ACM 是免费的。有关的信息 ACM，请参阅 [《AWS Certificate Manager 用户指南》](#)。

获取服务器证书

您可以通过调用 `IamClient`'s `getServerCertificate` 方法来检索服务器证书，然后将其 [GetServerCertificateRequest](#)与证书名称一起传递。

导入

```
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;
```

代码

```
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);
    }
}
```

请参阅上的[完整示例](#) GitHub。

列出服务器证书

要列出您的服务器证书，请使用调用 `IamClient`'s `listServerCertificates` 方法 [ListServerCertificatesRequest](#)。它返回 [ListServerCertificatesResponse](#)。

调用返回 `ListServerCertificateResponse` 对象的 `serverCertificateMetadataList` 方法以获取可用于获取有关每个证书的信息的 [ServerCertificateMetadata](#) 对象列表。

如果 `ListServerCertificateResponse` 对象的 `isTruncated` 方法返回了 `true`，调用 `ListServerCertificatesResponse` 对象的 `marker` 方法并使用标记创建一个新请求，则结果可能被截断。使用该新请求重新调用 `listServerCertificates` 以获取下一批结果。

导入

```
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;
```

代码

```
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);
    }
}
```

请参阅上的[完整示例](#) GitHub。

更新服务器证书

您可以通过调用's `updateServerCertificate` 方法来更新服务器证书 `IamClient`的名称或路径。它需要一个包含服务器证书当前名称的[UpdateServerCertificateRequest](#)对象集以及要使用的新名称或新路径。

导入

```
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;
```

代码

```
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);
    }
}
```

请参阅上的[完整示例](#) GitHub。

删除服务器证书

要删除服务器证书，请使用[DeleteServerCertificateRequest](#)包含证书名称 IamClient 的's deleteServerCertificate 方法进行调用。

导入

```
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;
```

代码

```
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);
    }
}
```

请参阅上的[完整示例](#) GitHub。

更多信息

- 在《IAM 用户指南》中使用@@ [服务器证书](#)
- [GetServerCertificate](#)在 IAM API 参考中
- [ListServerCertificates](#)在 IAM API 参考中
- [UpdateServerCertificate](#)在 IAM API 参考中
- [DeleteServerCertificate](#)在 IAM API 参考中

- [AWS Certificate Manager 用户指南](#)

与... 一起工作 Kinesis

本节提供[Amazon Kinesis](#)使用 AWS SDK for Java 2.x 进行编程的示例。

有关的更多信息 Kinesis，请参阅《[Amazon Kinesis 开发人员指南](#)》。

以下示例仅包含演示每种方法所需的代码。[完整的示例代码可在上找到 GitHub](#)。您可以从中下载单个源文件，也可以将存储库复制到本地以获得所有示例，然后构建并运行它们。

主题

- [订阅 Amazon Kinesis Data Streams](#)

订阅 Amazon Kinesis Data Streams

以下示例演示如何使用 `subscribeToShard` 方法检索和处理 Amazon Kinesis Data Streams 中的数据。Kinesis Data Streams 现在采用增强的扇出功能和低延迟 HTTP/2 数据检索 API，同时让开发人员能够更易于在同一 Kinesis 数据流上运行多个低延迟、高性能的应用程序。

设置

首先，创建一个异步 Kinesis 客户端和一个 [SubscribeToShardRequest](#) 对象。这些对象用于以下每个示例中以订阅 Kinesis 事件。

导入

```
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;
```

代码

```

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();

```

使用 Builder 接口

您可以使用 `builder` 方法来简化创建 [SubscribeToShardResponseHandler](#) 的过程。

使用生成器，您可以通过方法调用设置每个生命周期回调，而非实施完整的接口。

代码

```

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);
}

```

要对发布者进行更多控制，您可以使用 `publisherTransformer` 方法自定义发布者。

代码

```

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);
    }

```

请参阅 GitHub 上的[完整示例](#)。

使用自定义响应处理程序

要完全控制订阅用户和发布者，请实施 `SubscribeToShardResponseHandler` 接口。

在本示例中，您实施 `onEventStream` 方法，此方法允许您对发布者具有完全访问权限。此示例演示如何将发布者转换为事件记录以供订阅者打印。

代码

```

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);
}

```

请参阅 GitHub 上的[完整示例](#)。

使用 Visitor 接口

您可以使用 [Visitor](#) 对象订阅您有兴趣观看的特定事件。

代码

```

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);
}
```

请参阅 GitHub 上的[完整示例](#)。

使用自定义订阅者

您也可以实施您自己的自定义订阅者以订阅流。

此代码段显示了一个示例订阅者。

代码

```
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");
}
}
```

您可以将自定义订阅者传递给 `subscribe` 方法，如以下代码片段所示。

代码

```
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);
}
```

请参阅 GitHub 上的[完整示例](#)。

将数据记录写入 Kinesis 数据流

您可以使用 [KinesisClient](#) 对象通过 `putRecords` 方法将数据记录写入 Kinesis 数据流。要成功调用此方法，请创建一个 [PutRecordsRequest](#) 对象。将数据流的名称传递给 `streamName` 方法。此外，您必须使用 `putRecords` 方法传递数据（如下面的代码示例所示）。

导入

```
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;
```

在以下 Java 代码示例中，请注意 `StockTrade` 对象用作写入 Kinesis 数据流的数据。在运行此示例之前，请确保已创建数据流。

代码

```
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);
        }
    }
}
```

请参阅 GitHub 上的[完整示例](#)。

使用第三方库

您可以使用其他第三方库，而不是实现自定义订阅者。本示例演示了如何使用 RxJava 实现，但您可以使用实现反应式流接口的任何库。有关该库的更多信息，请参阅[Github 上的 RxJava 维基页面](#)。

要使用该库，请将其作为依赖项添加。如果您使用 Maven，示例将显示要使用的 POM 代码段。

POM 条目

```
<dependency>
  <groupId>io.reactivex.rxjava2</groupId>
  <artifactId>rxjava</artifactId>
  <version>2.1.14</version>
</dependency>
```

导入

```
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;
```

此示例在 `onEventStream` 生命周期方法中使用 RxJava。这样，您就对发布者具有完全访问权限，这可用于创建 Rx Flowable。

代码

```
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();
```

您也可以使用带有 `publisherTransformer` 发布者的 `Flowable` 方法。您必须使 `Flowable` 发布者适应 `SdkPublisher`，如以下示例所示。

代码

```
SubscribeToShardResponseHandler responseHandler =
SubscribeToShardResponseHandler
    .builder()
    .onError(t -> System.err.println("Error during stream - " +
t.getMessage()))
    .publisherTransformer(p ->
SdkPublisher.adapt(Flowable.fromPublisher(p).limit(100)))
    .build();
```

请参阅 GitHub 上的[完整示例](#)。

更多信息

- 《Amazon Kinesis API Reference》中的 [SubscribeToShardEvent](#)
- 《Amazon Kinesis API Reference》中的 [SubscribeToShard](#)

调用、列出和删除 AWS Lambda 函数

本节提供了使用 AWS SDK for Java 2.x 使用 Lambda 服务客户端进行编程的示例。

主题

- [调用 Lambda 函数](#)
- [列出 Lambda 函数](#)
- [删除 Lambda 函数](#)

调用 Lambda 函数

您可以通过创建 [LambdaClient](#) 对象并调用其 `invoke` 方法来调用 Lambda 函数。创建一个 [InvokeRequest](#) 对象以指定其他信息，例如要传递给函数的函数名称和有效负载。Lambda 函数

名称显示为 `arn: aws: lambda: us-east-1:123456789012: function:。HelloFunction` 可以通过查看 AWS Management Console 中的函数来检索值。

要将负载数据传递给函数，请创建一个包含信息的 [SdkBytes](#) 对象。例如，在以下代码示例中，请注意传递给 Lambda 函数的 JSON 数据。

导入

```
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;
```

代码

以下代码示例演示了如何调用 Lambda 函数。

```
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);
    }
}
```

请参阅上的[完整示例](#) GitHub。

列出 Lambda 函数

生成一个[Lambda Client](#)对象并调用其listFunctions方法。此方法返回一个[ListFunctionsResponse](#)对象。您可以调用此对象的functions方法来返回[FunctionConfiguration](#)对象列表。可以遍历该列表来检索有关函数的信息。例如，以下 Java 代码示例说明如何获取每个函数名称。

导入

```
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;
```

代码

以下 Java 代码示例演示如何检索 函数名称的列表。

```
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);
    }
}
```

请参阅上的[完整示例](#) GitHub。

删除 Lambda 函数

生成一个 [LambdaClient](#) 对象并调用其 `deleteFunction` 方法。创建一个 [DeleteFunctionRequest](#) 对象并将其传递给 `deleteFunction` 方法。此对象包含要删除的函数的名称等信息。函数名称显示为 `arn: aws: lambda: us-east-1:123456789012: function:。HelloFunction` 可以通过查看 AWS Management Console 中的函数来检索值。

导入

```
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;
```

代码

以下 Java 代码演示了如何删除 Lambda 函数。

```
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);
    }
}
```

请参阅上的 [完整示例](#) GitHub。

使用 Amazon S3

本部分提供使用 AWS SDK for Java 2.x 对 [Amazon Simple Storage Service \(S3 \)](#) 进行编程的示例。

以下示例仅包含演示每种方法所需的代码。 [完整的示例代码可在上找到 GitHub](#)。您可以从中下载单个源文件，也可以将存储库复制到本地以获得所有示例，然后构建并运行它们。

Note

从版本 2.18.x 及更高版本开始，在包括端点覆盖时 AWS SDK for Java 2.x 使用虚拟主机式寻址。只要存储桶名称是有效的DNS标签，这就适用。

在客户端生成器中使用 `true` 调用 [forcePathStyle](#) 方法，可强制客户端对桶使用路径式寻址。

以下示例显示了配置了端点覆盖并使用路径式寻址的服务客户端。

```
S3Client client = S3Client.builder()
    .region(Region.US_WEST_2)
    .endpointOverride(URI.create("https://s3.us-
west-2.amazonaws.com"))
    .forcePathStyle(true)
    .build();
```

使用接入点或多区域接入点

设置 [Amazon S3 接入点](#) 或 [多区域接入点](#) 后，您可以调用对象方法（例如 `putObject` 和 `getObject`），并提供接入点标识符而不是桶名称。

例如，如果接入点ARN标识符是 `arn:aws:s3:us-west-2:123456789012:accesspoint/test`，则可以使用以下代码段来调用该 `putObject` 方法。

```
Path path = Paths.get(URI.create("file:///temp/file.txt"));

s3Client.putObject(builder -> builder
    .key("myKey")
    .bucket("arn:aws:s3:us-west-2:123456789012:accesspoint/test")
    , path);
```

您也可以使用接入点的 [存储桶式别名](#) 来代替ARN字符串。 `bucket`

要使用多区域接入点，请将 `bucket` 参数替换为具有以下格式ARN的多区域接入点。

```
arn:aws:s3::account-id:accesspoint/MultiRegionAccessPoint_alias
```

使用SDK适用于 Java 的，添加以下 Maven 依赖关系以使用多区域接入点。在 Maven Central 中搜索 [latest version](#)。


```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>auth-crt</artifactId>
  <version>VERSION</version>
</dependency>
```

主题

- [创建、列出和删除 Amazon S3 桶](#)
- [使用 Amazon S3 对象](#)
- [使用 Amazon S3 预签名 URL](#)
- [Amazon S3 的跨区域访问](#)
- [带 Amazon S3 校验和](#)
- [使用高性能 S3 客户端：基于 AWS CRT 的 S3 客户端](#)
- [使用 Amazon S3 Transfer Manager 传输文件和目录](#)
- [使用 S3 事件通知](#)

创建、列出和删除 Amazon S3 桶

Amazon S3 中的每个对象（文件）必须位于存储桶中。存储桶表示对象的集合（容器）。每个存储桶必须具有一个唯一键（名称）。有关桶及其配置的详细信息，请参阅《Amazon Simple Storage Service 用户指南》中的[使用 Amazon S3 桶](#)。

Note

最佳实践

建议您对 [存储桶启用](#) AbortIncompleteMultipartUpload Amazon S3 生命周期规则。

该规则指示 Amazon S3 中止在启动后没有在指定天数内完成的分段上传。当超过设置的时间限制时，Amazon S3 将中止上传，然后删除未完成的上传数据。

有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的[使用版本控制的桶生命周期配置](#)。

Note

这些代码片段假设您了解基础知识，并且已使用 [the section called “设置 SDK 的单点登录访问权限”](#) 中的信息配置了默认 AWS 凭据。

创建存储桶

构建 [CreateBucketRequest](#) 并提供存储桶名称。将其传递到 S3Client 的 createBucket 方法。使用 S3Client 执行其他操作（例如列出或删除存储桶），如后面的示例中所示。

导入

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
```

代码

首先创建一个 S3Client。

```
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();
```

建立“创建存储桶请求”。

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
```

```
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class S3BucketOps {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        String bucket = "bucket" + System.currentTimeMillis();
        System.out.println(bucket);
        createBucket(s3, bucket);
        performOperations(s3, bucket);
    }

    // Create a bucket by using a S3Waiter object
    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                .bucket(bucketName)
                .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
                s3Waiter.waitUntilBucketExists(bucketRequestWait);
        }
    }
}
```

```
waiterResponse.matched().response().ifPresent(System.out::println);
System.out.println(bucketName + " is ready");

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

请参阅 GitHub 上的[完整示例](#)。

列出存储桶

构建 [ListBucketsRequest](#)。使用 S3Client 的 listBuckets 方法检索桶列表。如果请求成功，将返回 [ListBucketsResponse](#)。使用此响应对象可检索存储桶列表。

导入

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
```

代码

首先创建一个 S3Client。

```
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();
```

建立“列出存储桶请求”

```
// List buckets
ListBucketsRequest listBucketsRequest = ListBucketsRequest.builder().build();
ListBucketsResponse listBucketsResponse = s3.listBuckets(listBucketsRequest);
listBucketsResponse.buckets().stream().forEach(x ->
System.out.println(x.name()));
```

请参阅 GitHub 上的[完整示例](#)。

删除存储桶

在删除 Amazon S3 存储桶前，必须先确存储桶为空，否则该服务将返回错误。如果您的[存储桶受版本控制](#)，则必须同时删除位于存储桶中的所有受版本控制对象。

主题

- [删除桶中的对象](#)
- [删除空存储桶](#)

删除桶中的对象

构建 [ListObjectsV2Request](#) 并使用 S3Client 的 listObjects 方法检索桶中对象的列表。然后，在每个对象上使用 deleteObject 方法以删除它。

导入

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
```

代码

首先创建一个 S3Client。

```
ProfileCredentialsProvider credentialsProvider =
ProfileCredentialsProvider.create();
Region region = Region.US_EAST_1;
```

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

删除存储桶中的所有对象。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class S3BucketDeletion {
    public static void main(String[] args) throws Exception {
        final String usage = ""

            Usage:
                <bucket>

            Where:
                bucket - The bucket to delete (for example, bucket1).\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
```

```
        .region(region)
        .build();

deleteObjectsInBucket(s3, bucket);
s3.close();
}

public static void deleteObjectsInBucket(S3Client s3, String bucket) {
    try {
        // To delete a bucket, all the objects in the bucket must be deleted first.
        ListObjectsV2Request listObjectsV2Request = ListObjectsV2Request.builder()
            .bucket(bucket)
            .build();
        ListObjectsV2Response listObjectsV2Response;

        do {
            listObjectsV2Response = s3.listObjectsV2(listObjectsV2Request);
            for (S3Object s3Object : listObjectsV2Response.contents()) {
                DeleteObjectRequest request = DeleteObjectRequest.builder()
                    .bucket(bucket)
                    .key(s3Object.key())
                    .build();
                s3.deleteObject(request);
            }
        } while (listObjectsV2Response.isTruncated());
        DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder().bucket(bucket).build();
s3.deleteBucket(deleteBucketRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

请参阅 GitHub 上的[完整示例](#)。

删除空存储桶

使用桶名称构建 [DeleteBucketRequest](#) 并将其传递到 S3Client 的 deleteBucket 方法。

导入

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.ListBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
```

代码

首先创建一个 S3Client。

```
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();
```

删除存储桶。

```
DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
    .bucket(bucket)
    .build();

s3.deleteBucket(deleteBucketRequest);
s3.close();
```

请参阅 GitHub 上的[完整示例](#)。

使用 Amazon S3 对象

Amazon S3 对象表示一个文件或数据集合。每个对象都必须包含在一个[存储桶](#)中。

Note

最佳实践

建议您对 [存储桶启用](#) AbortIncompleteMultipartUpload Amazon S3 生命周期规则。

该规则指示 Amazon S3 中止在启动后没有在指定天数内完成的分段上传。当超过设置的时间限制时，Amazon S3 将中止上传，然后删除未完成的上传数据。
有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的[使用版本控制的桶生命周期配置](#)。

Note

这些代码片段假设您了解基础知识，并且已使用 [the section called “设置 SDK 的单点登录访问权限”](#) 中的信息配置了默认 AWS 凭据。

主题

- [上传对象](#)
- [分段上传对象](#)
- [删除对象](#)
- [列出对象](#)
- [更多示例](#)

上传对象

构建 [PutObjectRequest](#) 并提供存储桶名称和密钥名称。然后，将 S3Client 的 putObject 方法与包含对象内容和 PutObjectRequest 对象的 [RequestBody](#) 结合使用。存储桶必须存在，否则服务将返回错误。

导入

```
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;
```

代码

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

createBucket(s3, bucketName, region);

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

s3.putObject(objectRequest,
    RequestBody.fromByteBuffer(getRandomByteBuffer(10_000)));
```

请参阅 GitHub 上的[完整示例](#)。

分段上传对象

使用 `S3Client` 的 `createMultipartUpload` 方法获取上传 ID。然后，使用 `uploadPart` 方法上传每个段。最后，使用 `S3Client` 的 `completeMultipartUpload` 方法告知 Amazon S3 合并所有已上传的段并完成上传操作。

导入

```
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;
```

代码

```
        // First create a multipart upload and get the upload id
        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        CreateMultipartUploadResponse response =
s3.createMultipartUpload(createMultipartUploadRequest);
        String uploadId = response.uploadId();
        System.out.println(uploadId);

        // Upload all the different parts of the object
```

```
UploadPartRequest uploadPartRequest1 = UploadPartRequest.builder()
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)
    .partNumber(1).build();

String etag1 = s3
    .uploadPart(uploadPartRequest1,
RequestBody.fromByteBuffer(getRandomByteBuffer(5 * mB)))
    .eTag();

CompletedPart part1 =
CompletedPart.builder().partNumber(1).eTag(etag1).build();

UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder().bucket(bucketName).key(key)
    .uploadId(uploadId)
    .partNumber(2).build();

String etag2 = s3
    .uploadPart(uploadPartRequest2,
RequestBody.fromByteBuffer(getRandomByteBuffer(3 * mB)))
    .eTag();

CompletedPart part2 =
CompletedPart.builder().partNumber(2).eTag(etag2).build();

// Finally call completeMultipartUpload operation to tell S3 to merge
all

// uploaded
// parts and finish the multipart operation.
CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
    .parts(part1, part2)
    .build();

CompleteMultipartUploadRequest completeMultipartUploadRequest =
CompleteMultipartUploadRequest.builder()
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)
    .multipartUpload(completedMultipartUpload)
    .build();

s3.completeMultipartUpload(completeMultipartUploadRequest);
```

请参阅 GitHub 上的[完整示例](#)。

删除对象

构建 [DeleteObjectRequest](#) 并提供存储桶名称和密钥名称。使用 `S3Client` 的 `deleteObject` 方法，并向其传递要删除的桶和对象的名称。指定的存储桶和对象键必须存在，否则服务将返回错误。

导入

```
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;
```

代码

```
DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
    .bucket(bucketName)
    .key(key)
    .build();
```

```
s3.deleteObject(deleteObjectRequest);
```

请参阅 GitHub 上的[完整示例](#)。

复制对象

构建一个 [CopyObjectRequest](#)，并提供一个要将此对象负责复制到的存储桶名称、一个 URL 编码的字符串值（请参阅 `URLenCoder.code` 方法）以及该对象的密钥名称。使用 `S3Client` 的 `copyObject` 方法，并传递 [CopyObjectRequest](#) 对象。指定的存储桶和对象键必须存在，否则服务将返回错误。

导入

```
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;
```

代码

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CopyObject {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <objectKey> <fromBucket> <toBucket>
```

```
        Where:
            objectKey - The name of the object (for example, book.pdf).
            fromBucket - The S3 bucket name that contains the object (for
example, bucket1).
            toBucket - The S3 bucket to copy the object to (for example,
bucket2).
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String objectKey = args[0];
    String fromBucket = args[1];
    String toBucket = args[2];
    System.out.format("Copying object %s from bucket %s to %s\n", objectKey,
fromBucket, toBucket);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    copyBucketObject(s3, fromBucket, objectKey, toBucket);
    s3.close();
}

public static String copyBucketObject(S3Client s3, String fromBucket, String
objectKey, String toBucket) {
    CopyObjectRequest copyReq = CopyObjectRequest.builder()
        .sourceBucket(fromBucket)
        .sourceKey(objectKey)
        .destinationBucket(toBucket)
        .destinationKey(objectKey)
        .build();

    try {
        CopyObjectResponse copyRes = s3.copyObject(copyReq);
        return copyRes.copyObjectResult().toString();
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
        return "";
    }
}
```

请参阅 GitHub 上的[完整示例](#)。

列出对象

构建 [ListObjectsRequest](#) 并提供存储桶名称。然后，调用 S3Client 的 listObjects 方法并传递 ListObjectsRequest 对象。此方法返回 [ListObjectsResponse](#)，其中包含存储桶中的所有对象。您可以调用此对象的 contents 方法以获取对象的列表。您可以遍历这个列表来显示对象，如以下代码所示。

导入

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
import java.util.List;
```

代码

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListObjects {
```



```
public static void main(String[] args) {
    final String usage = ""

        Usage:
            <bucketName>\s

        Where:
            bucketName - The Amazon S3 bucket from which objects are read.\s
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    listBucketObjects(s3, bucketName);
    s3.close();
}

public static void listBucketObjects(S3Client s3, String bucketName) {
    try {
        ListObjectsRequest listObjects = ListObjectsRequest
            .builder()
            .bucket(bucketName)
            .build();

        ListObjectsResponse res = s3.listObjects(listObjects);
        List<S3Object> objects = res.contents();
        for (S3Object myValue : objects) {
            System.out.print("\n The name of the key is " + myValue.key());
            System.out.print("\n The object is " + calKb(myValue.size()) + " KBs");
            System.out.print("\n The owner is " + myValue.owner());
        }
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }  
  
    // convert bytes to kbs.  
    private static long calKb(Long val) {  
        return val / 1024;  
    }  
}
```

请参阅 GitHub 上的[完整示例](#)。

更多示例

本指南的[代码示例](#)部分包含更多使用 Amazon S3 对象的示例，包括如何[下载对象](#)。

使用 Amazon S3 预签名 URL

预签名 URL 提供对私有 S3 对象的临时访问权限，无需用户拥有 AWS 凭证或权限。

例如，假设 Alice 有权访问 S3 对象，并希望临时与 Bob 分享对该对象的访问权限。Alice 可以生成预签名的 GET 请求来与 Bob 分享，这样 Bob 就可以下载该对象而无需访问 Alice 的凭证。您可以为 HTTP GET 和 HTTP PUT 请求生成预签名 URL。

为对象生成预签名 URL，然后下载对象（GET 请求）

以下示例由两部分组成。

- 第 1 部分：Alice 为对象生成预签名 URL。
- 第 2 部分：Bob 使用预签名 URL 下载对象。

第 1 部分：生成 URL

Alice 在 S3 桶中已有一个对象。她使用以下代码生成一个 URL 字符串，Bob 可以在后续的 GET 请求中使用该字符串。

导入

```
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();
    }
}
```

第 2 部分：下载对象

Bob 使用以下三个代码选项之一来下载对象。或者，他可以使用浏览器来执行 GET 请求。

使用 JDK `URLConnection` (自 v1.1 起)

```
/* 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();
}
```

使用 JDK `HttpClient` (自 v1.1 起)

```
/* Use the JDK HttpClient (since v1.1) 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();
}

```

使用 SDK for Java 中的 **SdkHttpClient**

```

/* Use the AWS SDK for Java SdkHttpClient class to do the download. */
public byte[] useSdkHttpClientToPut(String presignedUrlString) {

    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream(); //
    Capture the response body to a byte array.
    try {
        URL presignedUrl = new URL(presignedUrlString);
        SdkHttpRequest request = SdkHttpRequest.builder()
            .method(SdkHttpMethod.GET)
            .uri(presignedUrl.toURI())
            .build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
            sdkHttpClient.prepareRequest(executeRequest).call();
            response.responseBody().ifPresentOrElse(
                abortableInputStream -> {
                    try {

```

```

        IoUtils.copy(abortableInputStream,
byteArrayOutputStream);
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
},
() -> logger.error("No response body."));

    logger.info("HTTP Response code is {}",
response.httpResponse().statusCode());
}
} catch (URISyntaxException | IOException e) {
    logger.error(e.getMessage(), e);
}
return byteArrayOutputStream.toByteArray();
}

```

请参阅 GitHub 上的[完整示例](#)和[测试](#)。

为上传生成预签名 URL，然后上传文件 (PUT 请求)

以下示例由两部分组成。

- 第 1 部分：Alice 生成用于上传对象的预签名 URL。
- 第 2 部分：Bob 使用预签名 URL 上传文件。

第 1 部分：生成 URL

Alice 已有一个 S3 桶。她使用以下代码生成一个 URL 字符串，Bob 可以在后续的 PUT 请求中使用该字符串。

导入

```

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();
    }
}

```

第 2 部分：上传文件对象

Bob 使用以下三个代码选项之一来上传文件。

使用 JDK `URLConnection` (自 v1.1 起)

```

/* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
public void useURLConnectionToPut(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);
    }
}
```

使用 JDK `HttpClient` (自 v11 起)

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

使用 SDK for Java 中的 `SdkHttpClient`

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

请参阅 GitHub 上的[完整示例](#)和[测试](#)。

Amazon S3 的跨区域访问

您在使用 Amazon Simple Storage Service (Amazon S3) 桶时，通常会知道桶的 AWS 区域。您使用的区域是在您创建 S3 客户端时确定的。

但有时，您可能需要使用一个特定的桶，但您不知道该桶是否位于为 S3 客户端设置的区域中。

您可以使用 SDK 启用跨不同区域访问 S3 桶的功能，而不必进行更多调用来确定桶区域。

设置

SDK 版本 2.20.111 已支持跨区域访问。请在 Maven 构建文件中使用此版本或更高版本作为 s3 依赖项，如以下代码段所示。

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>s3</artifactId>
  <version>2.20.111</version>
</dependency>
```

接下来，在创建 S3 客户端时，启用跨区域访问，如代码段所示。默认情况下，未启用此访问。

```
S3AsyncClient client = S3AsyncClient.builder()
    .crossRegionAccessEnabled(true)
    .build();
```

SDK 如何提供跨区域访问

当您在请求中引用现有桶时（例如使用 `putObject` 方法），SDK 会向为客户端配置的区域发起请求。

如果该特定区域中不存在该桶，则错误响应将包括该桶所在的实际区域。然后，SDK 在第二个请求中使用正确的区域。

为了优化将来对同一个桶的请求，SDK 会在客户端中缓存此区域映射。

注意事项

启用跨区域桶访问时，请注意，如果桶不在客户端配置的区域中，则可能会导致第一次 API 调用延迟增加。但是，后续调用会受益于缓存的区域信息，从而提高性能。

启用跨区域访问后，对桶的访问权限不会受到影响。无论桶位于哪个区域，用户都必须获得访问桶的授权。

带 Amazon S3 校验和

Amazon Simple Storage Service (Amazon S3) 允许您在上传对象时指定校验和。当您指定校验和时，校验和与对象一起存储，并且可以在下载对象时验证该校验和。

传输文件时，校验和可提供额外的数据层完整性。使用校验和，您可以通过确认收到文件与原始文件是否匹配来验证数据一致性。有关带 Amazon S3 的校验和的更多信息，请参阅 [Amazon Simple Storage Service 用户指南](#)。

Amazon S3 目前支持四种校验和算法：SHA-1、SHA-256、CRC-32 和 CRC-32C。您可以灵活地选择最适合自己的需求的算法，并让 SDK 计算校验和。或者，您可以使用其中四种支持算法之一来指定他们自己预先计算的校验和值。

我们在两个请求阶段讨论校验和：上传对象和下载对象。

上传对象

该算法的有效值为 CRC32、CRC32C、SHA1 和 SHA256。

以下代码片段展示了上传具有 CRC-32 校验和的对象的请求。当 SDK 发送此请求时，它会计算 CRC-32 校验和并上传对象。Amazon S3 将校验和与对象一起存储。

如果 SDK 计算的校验和与 Amazon S3 在收到请求时计算的校验和不匹配，则会返回错误。

使用预先计算的校验和值

与请求一起提供的预先计算校验和值会禁用 SDK 的自动计算，而是使用提供的值。

以下示例展示了具有预先计算的 SHA-256 校验和的请求。

如果 Amazon S3 确定指定算法的校验和值不正确，服务就会返回错误响应。

分段上传

您也可以将校验和用于分段上传。

下载对象

使用 [getObject](#) 方法下载对象时，，SDK 就会自动验证校验和。

以下代码段中的请求引导 SDK 通过计算校验和并比较值来验证响应中的校验和。

如果上传对象时没有使用校验和，则不会进行验证。

Amazon S3 中的一个对象可以具有多个校验和，但在下载时只验证一个校验和。以下优先顺序（基于校验和算法的效率）决定了 SDK 要验证哪个校验和：

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

例如，如果响应同时包含 CRC-32 和 SHA-256 校验和，则只验证 CRC-32 校验和。

使用高性能 S3 客户端：基于 AWS CRT 的 S3 客户端

基于 AWS CRT 的 S3 客户端（构建于 [AWS 通用运行时系统 \(CRT\) 之上](#)）是一种备选 S3 异步客户端。它通过自动使用 Amazon S3 的 [分段上传 API](#) 和 [字节范围提取](#)，将对象传入或传出 Amazon Simple Storage Service (Amazon S3)，从而提高了性能和可靠性。

基于 AWS CRT 的 S3 客户端可提高发生网络故障时的传输可靠性。通过重试文件传输中失败的各个分段，而无需从头开始重新启动传输，从而提高可靠性。

此外，基于 AWS CRT 的 S3 客户端还提供了增强的连接池和域名系统 (DNS) 负载平衡，这也提高了吞吐量。

您可以使用基于 AWS CRT 的 S3 客户端来代替 SDK 的标准 S3 异步客户端，并立即利用其提高的吞吐量。

SDK 中基于 AWS CRT 的组件

本主题中介绍的基于 AWS CRT 的 S3 客户端，与基于 AWS CRT 的 HTTP 客户端是 SDK 中的不同组件。

基于 AWS CRT 的 S3 客户端是 [S3AsyncClient](#) 接口的实现，用于与 Amazon S3 服务配合使用。它是基于 Java 的 [S3AsyncClient](#) 接口实现的替代方案，具有多种优势。

基于 [AWS CRT 的 HTTP 客户端](#) 是 [SdkAsyncHttpClient](#) 接口的实现，用于一般 HTTP 通信。它是 [Netty SdkAsyncHttpClient](#) 接口实现的替代方案，具有多种优势。

尽管两个组件都使用 [AWS 通用运行时系统](#) 中的库，但基于 AWS CRT 的 S3 客户端使用 [aws-c-s3 库](#)，并支持 [S3 分段上传 API](#) 功能。由于基于 AWS CRT 的 HTTP 客户端仅用于一般用途，因此它不支持 S3 分段上传 API 功能。

添加依赖项以使用基于 AWS CRT 的 S3 客户端

要使用基于 AWS CRT 的 S3 客户端，请将以下两个依赖项添加到您的 Maven 项目文件中。示例显示了要使用的最低版本。在 Maven Central 存储库中搜索 [s3](#) 和 [aws-crt](#) 构件的最新版本。

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>s3</artifactId>
  <version>2.20.68</version>
</dependency>
```

```
<dependency>
  <groupId>software.amazon.awssdk.crt</groupId>
  <artifactId>aws-crt</artifactId>
  <version>0.21.16</version>
</dependency>
```

创建基于 AWS CRT 的 S3 客户端的实例

使用默认设置创建基于 AWS CRT 的 S3 客户端的实例，如以下代码段所示。

```
S3AsyncClient s3AsyncClient = S3AsyncClient.crtCreate();
```

要配置客户端，请使用 AWS CRT 客户端生成器。您可以通过更改生成器方法，从标准 S3 异步客户端切换到基于 AWS CRT 的客户端。

```
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

AWS CRT 客户端生成器目前可能不支持标准生成器中的某些设置。通过调用 `S3AsyncClient#builder()` 获取标准生成器。

使用基于 AWS CRT 的 S3 客户端。

使用基于 AWS CRT 的 S3 客户端来调用 Amazon S3 API 操作。以下示例演示可通过 AWS SDK for Java 使用的 [PutObject](#) 和 [GetObject](#) 操作。

```
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();
```

使用 Amazon S3 Transfer Manager 传输文件和目录

Amazon S3 Transfer Manager 是针对 AWS SDK for Java 2.x 的一款开源高级文件传输工具。可以使用它将文件和目录传入和传出 Amazon Simple Storage Service (Amazon S3)。

在[AWS CRT 基于的 S3 客户端](#)之上构建时，S3 传输管理器可以利用性能改进，例如[分段上传 API](#)和[字节范围提取](#)。

使用 S3 Transfer Manager，您还可以实时监控传输进度，并暂停传输以便日后执行。

开始使用

将依赖项添加到构建文件中

要使用基于基于的 S3 客户端的性能增强的 S3 传输管理器，请使用以下依赖项配置您的构建文件。

AWS CRT

- 使用版本 **2.19.1** 或更高版本，SDK 适用于 Java 2.x。
- 将 `s3-transfer-manager` 构件添加为依赖项。
- 在版本中将 `aws-crt` 构件作为依赖项添加 **0.20.3** 或更高。

以下代码示例演示如何为 Maven 配置项目依赖项。

```
<project>
  <properties>
    <aws.sdk.version>2.19.1</aws.sdk.version>
  </properties>
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>${aws.sdk.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>s3-transfer-manager</artifactId>
    </dependency>
    <dependency>
      <groupId>software.amazon.awssdk.crt</groupId>
      <artifactId>aws-crt</artifactId>
      <version>0.20.3</version>
    </dependency>
  </dependencies>
</project>
```

在 Maven Central 存储库中搜索 [s3-transfer-manager](#) 和 [aws-crt](#) 构件的最新版本。

创建 S3 Transfer Manager 的实例

以下代码段显示了如何使用默认设置创建 [S3 TransferManager](#) 实例。

```
S3TransferManager transferManager = S3TransferManager.create();
```

以下示例演示如何使用自定义设置来配置 S3 Transfer Manager。在此示例中，使用[AWS CRT 基于 S3 的 S3 AsyncClient](#) 实例作为 S3 传输管理器的底层客户端。

```
S3AsyncClient s3AsyncClient = S3AsyncClient.crtBuilder()
    .credentialsProvider(DefaultCredentialsProvider.create())
```



```
.region(Region.US_EAST_1)
.targetThroughputInGbps(20.0)
.minimumPartSizeInBytes(8 * MB)
.build();

S3TransferManager transferManager = S3TransferManager.builder()
    .s3Client(s3AsyncClient)
    .build();
```

Note

如果构建文件中未包含aws-crt依赖关系，则 S3 传输管理器将建立在 for Java 2.x 中使用的标准 S3 异步客户端之SDK上。

将文件上传到 S3 桶

以下示例显示了文件上传示例 [LoggingTransferListener](#)，以及记录上传进度的可选用法。

要使用 S3 传输管理器将文件上传到 Amazon S3，请向S3TransferManager的[uploadFile](#)方法传递一个[UploadFileRequest](#)对象。

从该uploadFile方法返回的[FileUpload](#)对象表示上传过程。请求完成后，该[CompletedFileUpload](#)对象将包含有关上传的信息。

```
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();
}
```

导入

```
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;
```

从 S3 桶下载文件

以下示例显示了一个下载示例，以及记录下载进度的可选用法。 [LoggingTransferListener](#)

要使用 S3 传输管理器从 S3 存储桶下载对象，请生成一个 [DownloadFileRequest](#) 对象并将其传递给 [downloadFile](#) 方法。

的 `downloadFile` 方法返回 `S3TransferManager` 的 [FileDownload](#) 对象表示文件传输。下载完成后， [CompletedFileDownload](#) 包含对下载相关信息的访问权限。

```
public Long downloadFile(S3TransferManager transferManager, String bucketName,
                        String key, String downloadedFilePath) {
    DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
        .getObjectRequest(b -> b.bucket(bucketName).key(key))
        .destination(Paths.get(downloadedFilePath))
        .build();

    FileDownload downloadFile = transferManager.downloadFile(downloadFileRequest);

    CompletedFileDownload downloadResult = downloadFile.completionFuture().join();
    logger.info("Content length [{}]", downloadResult.response().contentLength());
    return downloadResult.response().contentLength();
}
```

导入

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
```

```
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;
```

将 Amazon S3 对象复制到另一个桶。

以下示例演示如何使用 S3 Transfer Manager 复制对象。

要开始将对象从 S3 存储桶复制到另一个存储桶，请创建一个基本 [CopyObjectRequest](#) 实例。

接下来，将基本内容封装 [CopyObjectRequest](#) 在 S3 传输管理器可以使用的文件中。 [CopyRequest](#)

[S3TransferManager](#) 的 `copy` 方法返回的 `Copy` 对象表示复制进程。复制过程完成后，该 [CompletedCopy](#) 对象将包含有关响应的详细信息。

```
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

要使用 S3 传输管理器执行跨区域复制，请在 AWS CRT 基于的 S3 客户端生成器 `crossRegionAccessEnabled` 上启用，如以下代码段所示。

```
S3AsyncClient s3AsyncClient = S3AsyncClient.crtBuilder()
    .crossRegionAccessEnabled(true)
    .build();

S3TransferManager transferManager = S3TransferManager.builder()
    .s3Client(s3AsyncClient)
    .build();
```

导入

```
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;
```

将本地目录上传到 S3 桶

以下示例演示如何将本地目录上传到 S3。

首先调用 `S3TransferManager` 实例 `uploadDirectory` 的方法，传入 `UploadDirectoryRequest`。

该 `DirectoryUpload` 对象表示上传过程，该过程会在请求完成 `CompletedDirectoryUpload` 时生成。 `CompletedDirectoryUpload` 对象包含有关传输结果的信息，包括哪些文件传输失败。

```
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();
}

```

导入

```

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;

```

将 S3 桶对象下载到本地目录

您可以将 S3 桶中的对象下载到本地目录，如以下示例所示。

要将 S3 存储桶中的对象下载到本地目录，请首先调用传输管理器的[downloadDirectory](#)方法，传入[DownloadDirectoryRequest](#)。

该[DirectoryDownload](#)对象表示下载过程，该过程会在请求完成[CompletedDirectoryDownload](#)时生成。CompleteDirectoryDownload 对象包含有关传输结果的信息，包括哪些文件传输失败。

```

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();
}
```

导入

```
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;
```

查看完整示例

[GitHub](#) 包含本页上所有示例的完整代码。

使用 S3 事件通知

为了帮助您监控存储桶中的活动，Amazon S3 可以在某些事件发生时发送通知。Amazon S3 用户指南提供了有关[存储桶可以发送的通知](#)的信息。

您可以使用 for Java 设置存储桶，将事件发送到四个可能SDK的目的地：

- Amazon Simple Notification Service 主题

- Amazon Simple Queue Service 队列
- AWS Lambda 函数
- Amazon EventBridge

当你设置要向其发送事件的存储桶时 EventBridge，你可以配置 EventBridge 一条规则，将同一个事件分散到多个目的地。当您存储桶配置为直接发送到前三个目的地之一时，只能为每个事件指定一种目标类型。

在下一节中，您将看到如何使用 SDK 适用于 Java 的配置存储桶，以便通过两种方式发送 S3 事件通知：直接发送到 Amazon SQS 队列和发送到 EventBridge。

最后一部分向您展示如何使用 S3 事件通知 API 以面向对象的方式处理通知。

将存储桶配置为直接发送到目的地

以下示例将存储桶配置为在针对存储桶发生对象创建事件或对象标记事件时发送通知。

```
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)
    );
}
```

上面显示的代码设置了一个队列来接收两种类型的事件。方便的是，该 `queueConfigurations` 方法允许您在需要时设置多个队列目的地。此外，在该 `notificationConfiguration` 方法中，您可以设置其他目标，例如一个或多个 Amazon SNS 主题或一个或多个 Lambda 函数。以下代码段显示了一个包含两个队列和三种目的地类型的示例。

```
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)
    );
```

代码示例 GitHub 存储库包含直接向队列发送 S3 事件通知的[完整示例](#)。

配置要发送到的存储桶 EventBridge

以下示例配置了要向其发送通知的存储桶。EventBridge

```
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());
}
```

在配置要向其发送事件的存储桶时 EventBridge，您只需指明 EventBridge 目的地，而不是事件的类型，也不是 EventBridge 要发送到的最终目的地。您可以使用 Java SDK 的 EventBridge 客户端配置最终目标和事件类型。

以下代码显示了如何配置 EventBridge 以将对象创建的事件分散到主题和队列。

```
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;
}
```

要在 Java 代码 EventBridge 中使用，请在 Maven pom.xml 文件中添加对 eventbridge 工件的依赖关系。

```
<dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>eventbridge</artifactId>
</dependency>
```

代码示例 GitHub 存储库包含向主题 EventBridge 和队列发送 S3 事件通知的[完整示例](#)，然后向主题和队列发送。

使用 S3 事件通知API来处理事件

目的地收到 S3 通知事件后，您可以使用 S3 事件通知以面向对象的方式处理这些事件。API您可以使用 S3 事件通知API来处理直接发送到目标的事件通知（如[第一个示例](#)所示），但不能处理通过 EventBridge发送的通知。存储桶发送的 S3 事件通知 EventBridge 包含 S3 事件通知当前API无法处理的不同结构。

添加依赖关系

S3 事件通知与SDK适用API于 Java 2.x 的 2.25.11 版本一起发布。

要使用 S3 事件通知API，请将所需的依赖项元素添加到您的 Maven 中pom.xml，如以下代码段所示。

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>bom</artifactId>
      <version>2.X.X1</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 [最新版本](#)。

使用该S3EventNotification课程

从JSON字符串创建S3EventNotification实例

要将JSON字符串转换为S3EventNotification对象，请使用该S3EventNotification类的静态方法，如下例所示。

```
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();

    }
}
```

在此示例中，该 `fromJson` 方法将 JSON 字符串转换为 `S3EventNotification` 对象。JSON 字符串中缺少字段将生成相应的 Java 对象字段中的 `null` 值，并且中任何多余的字段都JSON将被忽略。

事件通知记录的其他APIs内容可在API参考中找到[S3EventNotificationRecord](#)。

将**S3EventNotification**实例转换为字符JSON串

使用 `toJson` (或 `toJsonPretty`) 方法将 `S3EventNotification` 对象转换为字符JSON串，如以下示例所示。

```
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、ReplicationEventDataIntelligentTieringEventData、和LifecycleEventData的字段将被排除在外 (JSON如果是) null。其他null字段将序列化为。null

以下显示了 S3 对象标记事件的toJsonPretty方法输出示例。

```

{
  "Records" : [ {
    "eventVersion" : "2.3",
    "eventSource" : "aws:s3",
    "awsRegion" : "us-east-1",
    "eventTime" : "2024-07-19T20:09:18.551Z",
    "eventName" : "ObjectTagging:Put",
    "userIdentity" : {
      "principalId" : "AWS:XXXXXXXXXXXX"
    },
    "requestParameters" : {
      "sourceIPAddress" : "XXX.XX.XX.XX"
    },
    "responseElements" : {
      "x-amz-request-id" : "XXXXXXXXXXXX",
      "x-amz-id-2" : "XXXXXXXXXXXX"
    },
    "s3" : {
      "s3SchemaVersion" : "1.0",
      "configurationId" : "XXXXXXXXXXXX",
      "bucket" : {
        "name" : "DOC-EXAMPLE-BUCKET",
        "ownerIdentity" : {
          "principalId" : "XXXXXXXXXXXX"
        },
        "arn" : "arn:aws:s3:::XXXXXXXXXXXX"
      },
      "object" : {

```

```
        "key" : "akey",
        "size" : null,
        "eTag" : "XXXXXXXXXX",
        "versionId" : null,
        "sequencer" : null
    }
}
} ]
}
```

中提供了一个[完整的示例](#) GitHub，其中显示了如何使用API来处理亚马逊SQS队列收到的通知。

使用 Amazon Simple Notification Service

利用 Amazon Simple Notification Service，您可以通过多个通信通道轻松地将实时通知消息从您的应用程序推送给订阅者。本主题介绍如何执行 Amazon SNS 的一些基本功能。

创建主题

主题是通信通道的逻辑分组，可定义要将消息发送到的系统，例如，将消息发送到 AWS Lambda 和 HTTP Webhook。将消息发送到 Amazon SNS 之后，这些消息将分发到主题中定义的各个通道。这将使订阅者能够收到这些消息。

要创建主题，请先构建一个 [CreateTopicRequest](#) 对象，并使用构建器中的 `name()` 方法设置主题的名称。然后，使用 [SnsClient](#) 的 `createTopic()` 方法将请求对象发送到 Amazon SNS。可以将此请求的结果作为 [CreateTopicResponse](#) 对象捕获，如以下代码段中所示。

导入

```
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;
```

代码

```
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 "";
}
```

请参阅 GitHub 上的[完整示例](#)。

列出 Amazon SNS 主题

要检索现有 Amazon SNS 主题的列表，请构建一个 [ListTopicsRequest](#) 对象。然后，使用 `SnsClient` 的 `listTopics()` 方法将请求对象发送到 Amazon SNS。可以将此请求的结果作为 [ListTopicsResponse](#) 对象捕获。

以下代码段输出请求的 HTTP 状态代码以及您的 Amazon SNS 主题的 Amazon 资源名称 (ARN) 列表。

导入

```
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;
```

代码

```
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);
    }
}
```

请参阅 GitHub 上的[完整示例](#)。

为终端节点订阅主题

创建主题后，您可以配置将哪些通信通道作为该主题的终端节点。在 Amazon SNS 收到消息后，消息将分发给这些终端节点。

要将通信通道配置为主题的终端节点，请为该终端节点订阅主题。首先，请构建一个 [SubscribeRequest](#) 对象。将通信通道（例如，lambda 或 email）指定为 `protocol()`。将 `endpoint()` 设置为相关输出位置（例如，Lambda 函数的 ARN 或电子邮件地址），然后将要订阅的主题的 ARN 设置为 `topicArn()`。使用 `SnsClient` 的 `subscribe()` 方法将请求对象发送到 Amazon SNS。可以将此请求的结果作为 [SubscribeResponse](#) 对象捕获。

以下代码段说明如何为电子邮件地址订阅主题。

导入

```
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;
```

代码

```
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);
    }
}
```

请参阅 GitHub 上的[完整示例](#)。

向主题发布消息

如果您拥有一个主题并且已为该主题配置一个或多个终端节点，则可向该主题发布消息。首先，请构建一个 [PublishRequest](#) 对象。指定要发送的 `message()`，并指定要将消息发送到的主题的 ARN (`topicArn()`)。然后，使用 `SnsClient` 的 `publish()` 方法将请求对象发送到 Amazon SNS。可以将此请求的结果作为 [PublishResponse](#) 对象捕获。

导入

```
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;
```

代码

```
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);
    }
}
```

请参阅 GitHub 上的[完整示例](#)。

为终端节点取消订阅主题

可以删除配置为主题的终端节点的通信通道。执行此操作后，主题本身将继续存在，并将消息分发到为该主题配置的任何其他终端节点。

要删除作为主题的终端节点的通信通道，请为该终端节点取消订阅主题。首先，请构建一个 [UnsubscribeRequest](#) 对象，并将要取消订阅的主题的 ARN 设置为 `subscriptionArn()`。然后，使用 `SnsClient` 的 `unsubscribe()` 方法将请求对象发送到 SNS。可以将此请求的结果作为 [UnsubscribeResponse](#) 对象捕获。

导入

```
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;
```

代码

```
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);
    }
}
```

请参阅 GitHub 上的[完整示例](#)。

删除主题

要删除 Amazon SNS 主题，请先构建一个 [DeleteTopicRequest](#) 对象，并将主题的 ARN 设置为构建器中的 `topicArn()` 方法。然后，使用 `SnsClient` 的 `deleteTopic()` 方法将请求对象发送到 Amazon SNS。可以将此请求的结果作为 [DeleteTopicResponse](#) 对象捕获，如以下代码段中所示。

导入

```
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;
```

代码

```
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);
    }
}
```

请参阅 GitHub 上的[完整示例](#)。

有关更多信息，请参见 [Amazon Simple Notification Service 开发人员指南](#)。

与... 一起工作 Amazon Simple Queue Service

本节提供[Amazon Simple Queue Service](#)使用 AWS SDK for Java 2.x 进行编程的示例。

以下示例仅包含演示每种方法所需的代码。[完整的示例代码可在上找到 GitHub](#)。您可以从中下载单个源文件，也可以将存储库复制到本地以获得所有示例，然后构建并运行它们。

主题

- [使用 Amazon Simple Queue Service 消息队列](#)
- [发送、接收和删除 Amazon Simple Queue Service 消息](#)

使用 Amazon Simple Queue Service 消息队列

消息队列 是用于在 Amazon Simple Queue Service 中可靠地发送消息的逻辑容器。有两种类型的队列：标准 和先进先出 (FIFO)。要了解有关队列以及这些类型之间的差异的更多信息，请参阅《[Amazon Simple Queue Service Developer Guide](#)》。

本主题介绍如何使用AWS SDK for Java 来创建、列出、删除和获取 Amazon Simple Queue Service 队列的 URL。

以下示例中使用的 `sqsClient` 变量可以通过以下代码段创建。

```
SqsClient sqsClient = SqsClient.create();
```

当您使用静态`create()`方法创建`SqsClient`时，SDK 会使用默认区域[提供商链配置区域](#)，使用默认凭证[提供商链配置证书](#)。

创建队列

使用`SqsClient`的`createQueue`方法，并提供一个描述队列参数的[CreateQueueRequest](#)对象，如下代码片段所示。

导入

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
```

```
import java.util.List;
```

代码

```
CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
    .queueName(queueName)
    .build();

sqsClient.createQueue(createQueueRequest);
```

请查看[上面的完整示例](#) GitHub。

列出队列

要列出您账户的 Amazon Simple Queue Service 队列，请使用 [ListQueuesRequest](#) 对象调用 `SqsClient` 的 `listQueues` 方法。

当您使用不带任何参数的 [listQueues](#) 方法形式时，该服务会返回所有队列，最多 1,000 个队列。

您可以为 [ListQueuesRequest](#) 对象提供队列名称前缀，将结果限制为与该前缀匹配的队列，如以下代码所示。

导入

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

代码

```
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);  
    }  
}
```

请查看[上面的完整示例](#) GitHub。

获取队列的 URL

以下代码显示如何通过调用带有[GetQueueUrlRequest](#)对象的SqsClient'sgetQueueUrl方法来获取队列的 URL。

导入

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.sqs.SqsClient;  
import software.amazon.awssdk.services.sqs.model.*;  
import java.util.List;
```

代码

```
    GetQueueUrlResponse getQueueUrlResponse =  
  
    sqsClient.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());  
    String queueUrl = getQueueUrlResponse.queueUrl();  
    return queueUrl;
```

请查看[上面的完整示例](#) GitHub。

删除队列

提供队列指向[DeleteQueueRequest](#)对象的 [URL](#)。然后调用该SqsClient'sdeleteQueue方法删除队列，如以下代码所示。

导入

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.sqs.SqsClient;  
import software.amazon.awssdk.services.sqs.model.*;
```

```
import java.util.List;
```

代码

```
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);  
    }  
}
```

请查看[上面的完整示例](#) GitHub。

更多信息

- [CreateQueue](#)在 Amazon Simple Queue Service API 参考中
- [GetQueueUrl](#)在 Amazon Simple Queue Service API 参考中
- [ListQueues](#)在 Amazon Simple Queue Service API 参考中
- [DeleteQueue](#)在 Amazon Simple Queue Service API 参考中

发送、接收和删除 Amazon Simple Queue Service 消息

消息是可由分布式组件发送和接收的一段数据。始终使用 [SQS 队列](#) 发送消息。

以下示例中使用的 `sqsClient` 变量可以通过以下代码段创建。

```
SqsClient sqsClient = SqsClient.create();
```

当您使用静态 `create()` 方法创建 `SqsClient` 时，SDK 会使用默认区域 [提供商链配置区域](#)，使用默认凭证 [提供商链配置证书](#)。

发送消息

通过调用 `SqsClient` 客户端 `sendMessage` 方法向 Amazon Simple Queue Service 队列中添加一条消息。提供一个包含队列的 [URL](#)、消息正文和可选延迟值（以秒为单位）的 [SendMessageRequest](#) 对象。

导入

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

代码

```
sqsClient.sendMessage(SendMessageRequest.builder()
    .queueUrl(queueUrl)
    .messageBody("Hello world!")
    .delaySeconds(10)
    .build());

sqsClient.sendMessage(sendMsgRequest);
```

在一个请求中发送多条消息

通过使用 `SqsClient` `sendMessageBatch` 方法，在单个请求中发送多条消息。此方法采用 [SendMessageBatchRequest](#) 包含队列 URL 和要发送的消息列表的。（每条消息都是 [SendMessageBatchRequestEntry](#)。）您也可以通过设置消息上的延迟值来延迟发送特定消息。

导入

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

代码

```
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);
```

请查看[上面的完整示例](#) GitHub。

检索消息

通过调用 `SqsClient receiveMessage` 方法，检索当前位于队列中的任何消息。此方法采用 [ReceiveMessageRequest](#) 包含队列 URL 的。您也可以指定要返回的消息的最大数量。消息将作为一系列 [Message](#) 对象返回。

导入

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

代码

```
try {
    ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
    .queueUrl(queueUrl)
    .numberOfMessages(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;
}
```

请查看[上面的完整示例](#) GitHub。

收到后删除消息

收到消息并处理其内容后，通过向SqsClient的[deleteMessage](#)方法发送消息的接收句柄和队列URL，将该消息从队列中删除。

导入

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.*;
import java.util.List;
```

代码

```
try {
    for (Message message : messages) {
        DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .receiptHandle(message.receiptHandle())
                        .build();
        sqsClient.deleteMessage(deleteMessageRequest);
    }
}
```

请查看[上面的完整示例](#) GitHub。

更多信息

- 《Amazon Simple Queue Service Developer Guide》中的 [How Amazon Simple Queue Service Queues Work](#)
- [SendMessage](#)在 Amazon Simple Queue Service API 参考中
- [SendMessageBatch](#)在 Amazon Simple Queue Service API 参考中
- [ReceiveMessage](#)在 Amazon Simple Queue Service API 参考中
- [DeleteMessage](#)在 Amazon Simple Queue Service API 参考中

使用 Amazon Transcribe

以下示例显示如何使用 Amazon Transcribe 进行双向流式处理。双向流式处理意味着数据流同时转向服务且实时接收回来。此示例使用 Amazon Transcribe 流式转录发送音频流并实时收回转录的文本流。

要了解此功能的更多信息，请参阅《Amazon Transcribe Developer Guide》中的 [Streaming Transcription](#)。

要开始使用 Amazon Transcribe，请参阅《Amazon Transcribe Developer Guide》中的 [Getting Started](#)。

设置麦克风

此代码使用 javax.sound.sampled 软件包流式处理来自输入设备的音频。

代码

```
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;
    }
}
```

请参阅 GitHub 上的 [完整示例](#)。

创建发布者

该代码实现一个发布者，用于发布来自 Amazon Transcribe 音频流的音频数据。

代码

```
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();
}
}
}
```

请参阅 GitHub 上的[完整示例](#)。

创建客户端和启动流

在 main 方法中，创建请求对象，启动音频输入流，并通过音频输入实例化发布者。

您还必须创建 [StartStreamTranscriptionResponseHandler](#) 以指定如何处理来自 Amazon Transcribe 的响应。

然后，使用 TranscribeStreamingAsyncClient 的 startStreamTranscription 方法启动双向流式处理。

导入

```
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.StartStreamTranscriptionResponseHandler;
import software.amazon.awssdk.services.transcribestreaming.model.TranscriptEvent;
```

代码

```
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);
    }
}
```

请参阅 GitHub 上的[完整示例](#)。

更多信息

- 《Amazon Transcribe Developer Guide》中的[How It Works](#)。
- 《Amazon Transcribe Developer Guide》中的[Getting Started With Streaming Audio](#)。

SDK适用于 Java 2.x 的代码示例

本主题中的代码示例向您展示了如何使用 wit AWS SDK for Java 2.x h AWS。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

跨服务示例是指跨多个 AWS 服务工作的示例应用程序。

示例

- [用SDK于 Java 2.x 的操作和场景](#)
- [使用SDK适用于 Java 2.x 的跨服务示例](#)

用SDK于 Java 2.x 的操作和场景

以下代码示例展示了如何使用with来执行操作和实现常见场景 AWS 服务。 AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是指显示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

服务

- [APISDK适用于 Java 2.x 的网关示例](#)
- [适用SDK于 Java 2.x 的 Auto Scaling 示例](#)
- [用SDK于 Java 2.x 的应用程序恢复控制器示例](#)
- [用SDK于 Java 2.x 的 Aurora 示例](#)
- [使用SDK适用于 Java 2.x 的 Auto Scaling 示例](#)
- [用SDK于 Java 2.x 的 Amazon Bedrock 示例](#)
- [用SDK于 Java 2.x 的亚马逊 Bedrock 运行时示例](#)
- [CloudFront 用SDK于 Java 2.x 的示例](#)
- [CloudWatch 用SDK于 Java 2.x 的示例](#)

- [CloudWatch SDK适用于 Java 2.x 的事件示例](#)
- [CloudWatch 使用SDK适用于 Java 2.x 的日志示例](#)
- [SDK适用于 Java 2.x 的 Amazon Cognito 身份示例](#)
- [SDK适用于 Java 2.x 的 Amazon Cognito 身份提供商示例](#)
- [用于 Java 2.x 的 Amazon Comprehend SDK 示例](#)
- [使用适用于 Java 2.x 的 DynamoDB 示例 SDK](#)
- [使用SDK适用于 Java 2.x 的亚马逊EC2示例](#)
- [使用SDK适用于 Java 2.x 的亚马逊ECR示例](#)
- [使用SDK适用于 Java 2.x 的亚马逊ECS示例](#)
- [Elastic Load Balancing—— SDK 适用于 Java 2.x 的版本 2 示例](#)
- [MediaStore 用SDK于 Java 2.x 的示例](#)
- [OpenSearch SDK适用于 Java 2.x 的服务示例](#)
- [EventBridge 用SDK于 Java 2.x 的示例](#)
- [使用SDK适用于 Java 2.x 的 Forecast 示例](#)
- [AWS Glue 用SDK于 Java 2.x 的示例](#)
- [HealthImaging 用SDK于 Java 2.x 的示例](#)
- [IAM用SDK于 Java 2.x 的示例](#)
- [AWS IoT 用SDK于 Java 2.x 的示例](#)
- [AWS IoT data 用SDK于 Java 2.x 的示例](#)
- [SDK适用于 Java 2.x 的 Amazon Keyspaces 示例](#)
- [使用SDK适用于 Java 2.x 的 Kinesis 示例](#)
- [AWS KMS 用SDK于 Java 2.x 的示例](#)
- [用SDK于 Java 2.x 的 Lambda 示例](#)
- [AWS Marketplace SDK适用于 Java 2.x 的协议服务示例](#)
- [AWS Marketplace Catalog API 用SDK于 Java 2.x 的示例](#)
- [MediaConvert 用SDK于 Java 2.x 的示例](#)
- [使用SDK适用于 Java 2.x 的 Migration Hub 示例](#)
- [Amazon 使用SDK适用于 Java 2.x 的个性化示例](#)

- [使用SDK适用于 Java 2.x 的亚马逊个性化事件示例](#)
- [使用 Java 2.x 的 Amazon Person SDK alize 运行时示例](#)
- [使用SDK适用于 Java 2.x 的 Amazon Pinpoint 示例](#)
- [SDK适用于 Java 2.x 的 Amazon Pinpoint SMS 和 Voice API 示例](#)
- [使用SDK适用于 Java 2.x 的 Amazon Polly 示例](#)
- [使用SDK适用于 Java 2.x 的亚马逊RDS示例](#)
- [使用SDK适用于 Java 2.x 的 Amazon Redshift 示例](#)
- [使用适用于 Java 2.x 的亚马逊 Rekogn SDK ition 示例](#)
- [使用SDK适用于 Java 2.x 的 Route 53 域名注册示例](#)
- [SDK适用于 Java 2.x 的 Amazon S3 示例](#)
- [SDK适用于 Java 2.x 的 Amazon S3 控制示例](#)
- [用于 Java 2.x SDK 的 S3 Glacier 示例](#)
- [SageMaker 用SDK于 Java 2.x 的示例](#)
- [用SDK于 Java 2.x 的 Secrets Manager 示例](#)
- [使用SDK适用于 Java 2.x 的亚马逊SES示例](#)
- [使用SDK适用于 Java 2.x 的亚马逊 SES API v2 示例](#)
- [使用SDK适用于 Java 2.x 的亚马逊SNS示例](#)
- [使用SDK适用于 Java 2.x 的亚马逊SQS示例](#)
- [用SDK于 Java 2.x 的 Step Functions 示例](#)
- [AWS STS 用SDK于 Java 2.x 的示例](#)
- [AWS Support 用SDK于 Java 2.x 的示例](#)
- [使用SDK适用于 Java 2.x 的 Systems Manager 示例](#)
- [使用SDK适用于 Java 2.x 的 Amazon Textract 示例](#)
- [使用SDK适用于 Java 2.x 的 Amazon Transcribe 示例](#)

APISDK适用于 Java 2.x 的网关示例

以下代码示例向您展示了如何使用 with Gate API way 来执行操作和实现常见场景。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

CreateDeployment

以下代码示例演示如何使用 CreateDeployment。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateDeployment](#)”中的。

CreateRestApi

以下代码示例演示如何使用 CreateRestApi。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateRestApi](#)”中的。

DeleteDeployment

以下代码示例演示如何使用 DeleteDeployment。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteDeployment](#)”中的。

DeleteRestApi

以下代码示例演示如何使用 DeleteRestApi。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);  
    }  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteRestApi](#)”中的。

适用SDK于 Java 2.x 的 Auto Scaling 示例

以下代码示例向您展示了如何使用与 Application Auto Scaling AWS SDK for Java 2.x 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

DeleteScalingPolicy

以下代码示例演示如何使用 DeleteScalingPolicy。

SDK适用于 Java 2.x

Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ApplicationAutoScalingException;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DeleteScalingPolicyRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DeregisterScalableTargetRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsResponse;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesResponse;
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);
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteScalingPolicy](#)”中的。

RegisterScalableTarget

以下代码示例演示如何使用 RegisterScalableTarget。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
```

```
import
    software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient;
import
    software.amazon.awssdk.services.applicationautoscaling.model.ApplicationAutoScalingException;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsResponse;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequest;
import
    software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesResponse;
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.TargetTrackingScalingPolicyConfiguration;
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());
        }
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [RegisterScalableTarget](#)”中的。

用SDK于 Java 2.x 的应用程序恢复控制器示例

以下代码示例向您展示了如何通过 AWS SDK for Java 2.x 与应用程序恢复控制器一起使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

GetRoutingControlState

以下代码示例演示如何使用 GetRoutingControlState。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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;
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetRoutingControlState](#)”中的。

UpdateRoutingControlState

以下代码示例演示如何使用 UpdateRoutingControlState。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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;
    }
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [UpdateRoutingControlState](#)”中的。

用SDK于 Java 2.x 的 Aurora 示例

以下代码示例向您展示了如何通过 AWS SDK for Java 2.x 与 Aurora 一起使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

开始使用

开始使用 Aurora

以下代码示例显示如何开始使用 Aurora。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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()));
    }
}
```

- 有关API详细信息，请参阅describeDBClusters 《AWS SDK for Java 2.x API参考资料》中的 [D](#)。

主题

- [操作](#)
- [场景](#)

操作

CreateDBCluster

以下代码示例演示如何使用 CreateDBCluster。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";
}
```

- 有关API详细信息，请参阅[createDBCluster](#) 《AWS SDK for Java 2.x API参考资料》中的 [C](#)。

CreateDBClusterParameterGroup

以下代码示例演示如何使用 `CreateDBClusterParameterGroup`。

SDK适用于 Java 2.x

Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅[createDBClusterParameterGroup](#) 《AWS SDK for Java 2.x API参考资料》中的 [C](#)。

CreateDBClusterSnapshot

以下代码示例演示如何使用 CreateDBClusterSnapshot。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅《AWS SDK for Java 2.x API参考资料》中的 [CreateDBCluster 快照](#)。

CreateDBInstance

以下代码示例演示如何使用 CreateDBInstance。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";
}
```

- 有关API详细信息，请参阅createDBInstance 《AWS SDK for Java 2.x API参考资料》中的 [C](#)。

DeleteDBCluster

以下代码示例演示如何使用 DeleteDBCluster。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅deleteDBCluster 《AWS SDK for Java 2.x API参考资料》中的 [D](#)。

DeleteDBClusterParameterGroup

以下代码示例演示如何使用 DeleteDBClusterParameterGroup。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```

    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);
    }
}
```

- 有关API详细信息，请参阅`deleteDBClusterParameterGroup`《AWS SDK for Java 2.x API参考资料》中的 [D](#)。

DeleteDBInstance

以下代码示例演示如何使用 `DeleteDBInstance`。

SDK适用于 Java 2.x

Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅deleteDBInstance 《AWS SDK for Java 2.x API参考资料》中的 [D](#)。

DescribeDBClusterParameterGroups

以下代码示例演示如何使用 DescribeDBClusterParameterGroups。

SDK适用于 Java 2.x

Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅[describeDBClusterParameterGroups](#) 《AWS SDK for Java 2.x API参考资料》中的 [D](#)。

DescribeDBClusterParameters

以下代码示例演示如何使用 DescribeDBClusterParameters。

SDK适用于 Java 2.x

Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅《AWS SDK for Java 2.x API参考资料》中的 [DescribeDBCluster 参数](#)。

DescribeDBClusterSnapshots

以下代码示例演示如何使用 DescribeDBClusterSnapshots。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
}
}
```

- 有关API详细信息，请参阅《AWS SDK for Java 2.x API参考资料》中的 [DescribeDBCluster快照](#)。

DescribeDBClusters

以下代码示例演示如何使用 DescribeDBClusters。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
}
}
```

- 有关API详细信息，请参阅describeDBClusters 《AWS SDK for Java 2.x API参考资料》中的 [D](#)。

DescribeDBEngineVersions

以下代码示例演示如何使用 DescribeDBEngineVersions。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "
        + engine0b.dbParameterGroupFamily());
        System.out.println("The name of the database engine " +
engine0b.engine());
        System.out.println("The version number of the database engine " +
engine0b.engineVersion());
    }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅AWS SDK for Java 2.x API参考中的 [DescribeDBEngine 版本](#)。

DescribeDBInstances

以下代码示例演示如何使用 DescribeDBInstances。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

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

- 有关API详细信息，请参阅describeDBInstances 《AWS SDK for Java 2.x API参考资料》中的 [D](#)。

DescribeOrderableDBInstanceOptions

以下代码示例演示如何使用 DescribeOrderableDBInstanceOptions。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeOrderableDBInstanceOptions](#)”中的。

ModifyDBClusterParameterGroup

以下代码示例演示如何使用 ModifyDBClusterParameterGroup。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅[ModifyDBClusterParameterGroup](#) 《AWS SDK for Java 2.x API参考资料》中的 [M](#)。

场景

开始使用数据库集群

以下代码示例演示了操作流程：

- 创建自定义 Aurora 数据库集群参数组并设置参数值。
- 创建一个使用参数组的数据库集群。
- 创建包含数据库的数据库实例。
- 拍摄数据库集群的快照，然后清理资源。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including 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);
    }
}
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [C reateDBCluster](#)
 - [C reateDBCluster ParameterGroup](#)
 - [C reateDBCluster 快照](#)
 - [C reateDBInstance](#)
 - [D eleteDBCluster](#)
 - [D eleteDBCluster ParameterGroup](#)
 - [D eleteDBInstance](#)
 - [D escribeDBCluster ParameterGroups](#)
 - [D escribeDBCluster 参数](#)
 - [D escribeDBCluster 快照](#)
 - [D escribeDBClusters](#)
 - [D escribeDBEngine 版本](#)
 - [D escribeDBInstances](#)
 - [DescribeOrderableDBInstanceOptions](#)
 - [M odifyDBCluster ParameterGroup](#)

使用SDK适用于 Java 2.x 的 Auto Scaling 示例

以下代码示例向您展示了如何使用与 Auto Scaling AWS SDK for Java 2.x 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

开始使用

Hello Auto Scaling

以下代码示例显示如何开始使用自动扩缩。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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());
        });
    }
}
```

```
    });  
  }  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeAutoScalingGroups](#)”中的。

主题

- [操作](#)
- [场景](#)

操作

CreateAutoScalingGroup

以下代码示例演示如何使用 CreateAutoScalingGroup。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateAutoScalingGroup](#)”中的。

DeleteAutoScalingGroup

以下代码示例演示如何使用 DeleteAutoScalingGroup。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteAutoScalingGroup](#)”中的。

DescribeAutoScalingGroups

以下代码示例演示如何使用 DescribeAutoScalingGroups。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeAutoScalingGroups](#)”中的。

DescribeAutoScalingInstances

以下代码示例演示如何使用 DescribeAutoScalingInstances。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeAutoScalingInstances](#)”中的。

DescribeScalingActivities

以下代码示例演示如何使用 DescribeScalingActivities。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeScalingActivities](#)”中的。

DisableMetricsCollection

以下代码示例演示如何使用 `DisableMetricsCollection`。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DisableMetricsCollection](#)”中的。

EnableMetricsCollection

以下代码示例演示如何使用 EnableMetricsCollection。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [EnableMetricsCollection](#)”中的。

SetDesiredCapacity

以下代码示例演示如何使用 SetDesiredCapacity。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [SetDesiredCapacity](#)”中的。

TerminateInstanceInAutoScalingGroup

以下代码示例演示如何使用 TerminateInstanceInAutoScalingGroup。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [TerminateInstanceInAutoScalingGroup](#)”中的。

UpdateAutoScalingGroup

以下代码示例演示如何使用 UpdateAutoScalingGroup。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [UpdateAutoScalingGroup](#)”中的。

场景

构建和管理弹性服务

以下代码示例演示了如何创建可返回书籍、电影和歌曲推荐的负载均衡的 Web 服务。该示例演示服务如何响应故障，以及如何重组服务以提高故障发生时的弹性。

- 使用 Amazon A EC2 uto Scaling 组根据启动模板创建亚马逊弹性计算云 (AmazonEC2) 实例，并将实例数量保持在指定范围内。
- 使用 Elastic Load Balancing 处理和分发HTTP请求。

- 监控自动扩缩组中实例的运行状况，并仅将请求转发到运行状况良好的实例。
- 在每个EC2实例上运行 Python 网络服务器来处理HTTP请求。Web 服务器以建议和运行状况检查作为响应。
- 使用 Amazon DynamoDB 表模拟推荐服务。
- 通过更新 AWS Systems Manager 参数来控制 Web 服务器对请求和运行状况检查的响应。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

在命令提示符中运行交互式场景。

```
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 wasSuccessful = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
if (!wasSuccessful) {
    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 (wasSuccessful) {
    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 =
loadBalancer.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("""
check to update
        Note that it can take a minute or two for the health
        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);
}
}

```

创建一个包含 Auto Scaling 和 Amazon EC2 操作的类。

```

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 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.getInstanceInformationList();
        for (InstanceInformation info : instanceInformationList) {
            if (info.getInstanceId().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.");
}
}

```

创建一个包含弹性负载均衡操作的类。

```

public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                .region(Region.US_EAST_1)
                .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
            .names(targetGroupName)
            .build();
    }
}

```

```
DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
    .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
    .build();

DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
return healthResponse.targetHealthDescriptions();
}

// Gets the HTTP endpoint of the load balancer.
public String getEndpoint(String lbName) {
    DescribeLoadBalancersResponse res = getLoadBalancerClient()
        .describeLoadBalancers(describe -> describe.names(lbName));
    return res.loadBalancers().get(0).dnsName();
}

// Deletes a load balancer.
public void deleteLoadBalancer(String lbName) {
    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
            .build();

        getLoadBalancerClient().deleteLoadBalancer(
            builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancersDeleted(request);
        waiterResponse.matched().response().ifPresent(System.out::println);

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.out.println(lbName + " was deleted.");
    }

    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                .describeTargetGroups(describe ->
describe.names(targetGroupName));
            getLoadBalancerClient()
                .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }

    // Verify this computer can successfully send a GET request to the load balancer
    // endpoint.
    public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
        boolean success = false;
        int retries = 3;
        CloseableHttpClient httpClient = HttpClients.createDefault();

        // Create an HTTP GET request to the ELB.
        HttpGet httpGet = new HttpGet("http://" + elbDnsName);
        try {
            while ((!success) && (retries > 0)) {
                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);
                if (statusCode == 200) {
                    success = true;
                } else {
                    retries--;
                    System.out.println("Got connection error from load balancer
endpoint, retrying...");
                    TimeUnit.SECONDS.sleep(15);
                }
            }
        }
    }
}
```

```
    } catch (org.apache.http.conn.HttpHostConnectException e) {
        System.out.println(e.getMessage());
    }

    System.out.println("Status.." + success);
    return success;
}

/**
 * Creates an Elastic Load Balancing target group. The target group specifies
 * how
 * the load balancer forward requests to instances in the group and how instance
 * health is checked.
 */
public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
        .name(targetGroupName)
        .protocol(protocol)
        .build();

    CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
    String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
    String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
    System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
    return targetGroupArn;
}

/**
 * Creates an Elastic Load Balancing load balancer that uses the specified
 * subnets
 * and forwards requests to the specified target group.
 */
public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
```



```
String protocol) {
    try {
        List<String> subnetIdStrings = subnetIds.stream()
            .map(Subnet::subnetId)
            .collect(Collectors.toList());

        CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
            .subnets(subnetIdStrings)
            .name(lbName)
            .scheme("internet-facing")
            .build();

        // Create and wait for the load balancer to become available.
        CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
        String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(lbARN)
            .build();

        System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

        // Create a listener for the load balance.
        Action action = Action.builder()
            .targetGroupArn(targetGroupARN)
            .type("forward")
            .build();

        CreateListenerRequest listenerRequest = CreateListenerRequest.builder()
```

```

        .loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
            .defaultActions(action)
            .port(port)
            .protocol(protocol)
            .defaultActions(action)
            .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
            + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
}

```

创建一个使用 DynamoDB 模拟推荐服务的类。

```

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.waitForTableExists(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);
}
}

```

创建一个包含 Systems Manager 操作的类。

```

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);
}
}
```

• 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。

- [AttachLoadBalancerTargetGroups](#)
- [CreateAutoScalingGroup](#)
- [CreateInstanceProfile](#)
- [CreateLaunchTemplate](#)
- [CreateListener](#)
- [CreateLoadBalancer](#)
- [CreateTargetGroup](#)
- [DeleteAutoScalingGroup](#)
- [DeleteInstanceProfile](#)
- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeIamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)

- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplacelamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

管理组和实例

以下代码示例展示了如何：

- 使用启动模板和可用区创建一个 Amazon A EC2 uto Scaling 群组，并获取有关正在运行的实例的信息。
- 启用 Amazon CloudWatch 指标收集。
- 更新组的所需容量，并等待实例启动。
- 终止组中的实例。
- 列出为响应用户请求和容量变化而发生的扩缩活动。
- 获取 CloudWatch 指标的统计数据，然后清理资源。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```

* In addition, create a launch template. For more information, see the
* following topic:
*
* https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-templates.html#create-launch-template
*
* This code example performs the following operations:
* 1. Creates an Auto Scaling group using an AutoScalingWaiter.
* 2. Gets a specific Auto Scaling group and returns an instance Id value.
* 3. Describes Auto Scaling with the Id value.
* 4. Enables metrics collection.
* 5. Update an Auto Scaling group.
* 6. Describes Account details.
* 7. Describe account details"
* 8. Updates an Auto Scaling group to use an additional instance.
* 9. Gets the specific Auto Scaling group and gets the number of instances.
* 10. List the scaling activities that have occurred for the group.
* 11. Terminates an instance in the Auto Scaling group.
* 12. Stops the metrics collection.
* 13. Deletes the Auto Scaling group.
*/

public class AutoScalingScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = ""

            Usage:
                <groupName> <launchTemplateName> <vpcZoneId>

            Where:
                groupName - The name of the Auto Scaling group.
                launchTemplateName - The name of the launch template.\s
                vpcZoneId - A subnet Id for a virtual private cloud (VPC) where
instances in the Auto Scaling group can be created.
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String groupName = args[0];

```



```
String launchTemplateName = args[1];
String vpcZoneId = args[2];
AutoScalingClient autoScalingClient = AutoScalingClient.builder()
    .region(Region.US_EAST_1)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the Amazon EC2 Auto Scaling example
scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create an Auto Scaling group named " + groupName);
createAutoScalingGroup(autoScalingClient, groupName, launchTemplateName,
vpcZoneId);
System.out.println(
    "Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned");
Thread.sleep(60000);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Get Auto Scale group Id value");
String instanceId = getSpecificAutoScalingGroups(autoScalingClient,
groupName);
if (instanceId.compareTo("") == 0) {
    System.out.println("Error - no instance Id value");
    System.exit(1);
} else {
    System.out.println("The instance Id value is " + instanceId);
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Describe Auto Scaling with the Id value " +
instanceId);
describeAutoScalingInstance(autoScalingClient, instanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Enable metrics collection " + instanceId);
enableMetricsCollection(autoScalingClient, groupName);
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("5. Update an Auto Scaling group to update max size to
3");
updateAutoScalingGroup(autoScalingClient, groupName, launchTemplateName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Describe Auto Scaling groups");
describeAutoScalingGroups(autoScalingClient, groupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Describe account details");
describeAccountLimits(autoScalingClient);
System.out.println(
    "Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned");
Thread.sleep(60000);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Set desired capacity to 2");
setDesiredCapacity(autoScalingClient, groupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Get the two instance Id values and state");
getSpecificAutoScalingGroups(autoScalingClient, groupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. List the scaling activities that have occurred for
the group");
describeScalingActivities(autoScalingClient, groupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Terminate an instance in the Auto Scaling group");
terminateInstanceInAutoScalingGroup(autoScalingClient, instanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Stop the metrics collection");
disableMetricsCollection(autoScalingClient, groupName);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Delete the Auto Scaling group");
        deleteAutoScalingGroup(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);

        autoScalingClient.close();
    }

    public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
                .autoScalingGroupName(groupName)
                .maxRecords(10)
                .build();

            DescribeScalingActivitiesResponse response = autoScalingClient
                .describeScalingActivities(scalingActivitiesRequest);
            List<Activity> activities = response.activities();
            for (Activity activity : activities) {
                System.out.println("The activity Id is " + activity.activityId());
                System.out.println("The activity details are " +
activity.details());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
                .autoScalingGroupName(groupName)
```

```
        .desiredCapacity(2)
        .build();

        autoScalingClient.setDesiredCapacity(capacityRequest);
        System.out.println("You have set the DesiredCapacity to 2");

    } catch (AutoScalingException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createAutoScalingGroup(AutoScalingClient autoScalingClient,
    String groupName,
    String launchTemplateName,
    String vpcZoneId) {
    try {
        AutoScalingWaiter waiter = autoScalingClient.waiter();
        LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
            .launchTemplateName(launchTemplateName)
            .build();

        CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .availabilityZones("us-east-1a")
            .launchTemplate(templateSpecification)
            .maxSize(1)
            .minSize(1)
            .vpcZoneIdentifier(vpcZoneId)
            .build();

        autoScalingClient.createAutoScalingGroup(request);
        DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
            .autoScalingGroupNames(groupName)
            .build();

        WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
            .waitUntilGroupExists(groupsRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Auto Scaling Group created");
    }
}
```

```
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
        try {
            DescribeAutoScalingInstancesRequest describeAutoScalingInstancesRequest
= DescribeAutoScalingInstancesRequest
                .builder()
                .instanceIds(id)
                .build();

            DescribeAutoScalingInstancesResponse response = autoScalingClient

.describeAutoScalingInstances(describeAutoScalingInstancesRequest);
            List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
            for (AutoScalingInstanceDetails instance : instances) {
                System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void describeAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .maxRecords(10)
                .build();

            DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups(groupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
```

```
        for (AutoScalingGroup group : groups) {
            System.out.println("*** The service to use for the health checks: "
+ group.healthCheckType());
        }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getSpecificAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            String instanceId = "";
            DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

            DescribeAutoScalingGroupsResponse response = autoScalingClient
                .describeAutoScalingGroups(scalingGroupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
                System.out.println("The group name is " +
group.autoScalingGroupName());
                System.out.println("The group ARN is " +
group.autoScalingGroupARN());
                List<Instance> instances = group.instances();

                for (Instance instance : instances) {
                    instanceId = instance.instanceId();
                    System.out.println("The instance id is " + instanceId);
                    System.out.println("The lifecycle state is " +
instance.lifecycleState());
                }
            }

            return instanceId;
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

```
    }

    public static void enableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
                .autoScalingGroupName(groupName)
                .metrics("GroupMaxSize")
                .granularity("1Minute")
                .build();

            autoScalingClient.enableMetricsCollection(collectionRequest);
            System.out.println("The enable metrics collection operation was
successful");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void disableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
                .autoScalingGroupName(groupName)
                .metrics("GroupMaxSize")
                .build();

            autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
            System.out.println("The disable metrics collection operation was
successful");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void describeAccountLimits(AutoScalingClient autoScalingClient) {
        try {
```

```
        DescribeAccountLimitsResponse response =
autoScalingClient.describeAccountLimits();
        System.out.println("The max number of auto scaling groups is " +
response.maxNumberOfAutoScalingGroups());
        System.out.println("The current number of auto scaling groups is " +
response.numberOfWorkAutoScalingGroups());

    } 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);
    }
}
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。

- [CreateAutoScalingGroup](#)
- [DeleteAutoScalingGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAutoScalingInstances](#)
- [DescribeScalingActivities](#)
- [DisableMetricsCollection](#)
- [EnableMetricsCollection](#)
- [SetDesiredCapacity](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

用SDK于 Java 2.x 的 Amazon Bedrock 示例

以下代码示例向您展示了如何使用 AWS SDK for Java 2.x 与 Amazon Bedrock 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

GetFoundationModel

以下代码示例演示如何使用 `GetFoundationModel`。

SDK适用于 Java 2.x

Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用同步 Amazon Bedrock 客户端获取有关基础模型的详细信息。

```
/**
 * 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);
    }
}

```

使用异步 Amazon Bedrock 客户端获取有关基础模型的详细信息。

```

/**
 * 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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetFoundationModel](#)”中的。

ListFoundationModels

以下代码示例演示如何使用 ListFoundationModels。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用同步 Amazon Bedrock 客户端列出可用的 Amazon Bedrock 基础模型。

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

使用异步 Amazon Bedrock 客户端列出可用的 Amazon Bedrock 基础模型。

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

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListFoundationModels](#)”中的。

用SDK于 Java 2.x 的亚马逊 Bedrock 运行时示例

以下代码示例向您展示了如何使用 AWS SDK for Java 2.x 与 Amazon Bedrock Runtime 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [AI21实验室侏罗纪-2](#)
- [Amazon Titan Image Generator](#)
- [亚马逊 Titan 文本](#)
- [Amazon Titan Text Embeddings](#)
- [Anthropic Claude](#)
- [Cohere Command](#)
- [Meta Llama](#)
- [Mistral AI](#)
- [场景](#)
- [Stable Diffusion](#)

AI21实验室侏罗纪-2

交谈

以下代码示例展示了如何使用 Bedrock 的 Converse 向 AI21 Labs Jurassic-2 发送短信。API

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用 Bedrock 的 Converse 向 AI21 Labs Jurassic-2 发送短信。API

```
// Use the Converse API to send a text message to AI21 Labs Jurassic-2.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Jurassic-2 Mid.
        var modelId = "ai21.j2-mid-v1";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                    .maxTokens(512)
```

```
        .temperature(0.5F)
        .topP(0.9F)));

    // Retrieve the generated text from Bedrock's response object.
    var responseText = response.output().message().content().get(0).text();
    System.out.println(responseText);

    return responseText;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}

}

public static void main(String[] args) {
    converse();
}
}
```

使用 Bedrock 的 Converse 和异步 Java 客户端，向 AI21 Labs Jurassic-2 发送短信。

```
// Use the Converse API to send a text message to AI21 Labs Jurassic-2
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
```

```
// Replace the DefaultCredentialsProvider with your preferred credentials
provider.
var client = BedrockRuntimeAsyncClient.builder()
    .credentialsProvider(DefaultCredentialsProvider.create())
    .region(Region.US_EAST_1)
    .build();

// Set the model ID, e.g., Jurassic-2 Mid.
var modelId = "ai21.j2-mid-v1";

// Create the input text and embed it in a message object with the user
role.
var inputText = "Describe the purpose of a 'hello world' program in one
line.";
var message = Message.builder()
    .content(ContentBlock.fromText(inputText))
    .role(ConversationRole.USER)
    .build();

// Send the message with a basic inference configuration.
var request = client.converse(params -> params
    .modelId(modelId)
    .messages(message)
    .inferenceConfig(config -> config
        .maxTokens(512)
        .temperature(0.5F)
        .topP(0.9F))
);

// Prepare a future object to handle the asynchronous response.
CompletableFuture<String> future = new CompletableFuture<>();

// Handle the response or error using the future object.
request.whenComplete((response, error) -> {
    if (error == null) {
        // Extract the generated text from Bedrock's response object.
        String responseText =
response.output().message().content().get(0).text();
        future.complete(responseText);
    } else {
        future.completeExceptionally(error);
    }
});
```

```
        try {
            // Wait for the future object to complete and retrieve the generated
            text.

            String responseText = future.get();
            System.out.println(responseText);

            return responseText;

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        converseAsync();
    }
}
```

- 有关API详细信息，请参阅《AWS SDK for Java 2.x API参考资料》中的“[Converse](#)”。

InvokeModel

以下代码示例演示如何使用调用模型向 AI21 Labs Jurassic-2 发送短信。API

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用调用模型API发送短信。

```
// Use the native inference API to send a text message to AI21 Labs Jurassic-2.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
```

```
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Jurassic-2 Mid.
        var modelId = "ai21.j2-mid-v1";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-jurassic2.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{prompt}}\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in the model's native request payload.
        String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                .body(SdkBytes.fromUtf8String(nativeRequest))
                .modelId(modelId)
            );

            // Decode the response body.
            var responseBody = new JSONObject(response.body().asUtf8String());

            // Retrieve the generated text from the model's response.
```

```
        var text = new JSONPointer("/completions/0/data/text").queryFrom(responseBody).toString();
        System.out.println(text);

        return text;

    } catch (SdkClientException e) {
        System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
            e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    invokeModel();
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [InvokeModel](#)”中的。

Amazon Titan Image Generator

InvokeModel

以下代码示例展示了如何在 Amazon Bedrock 上调用 Amazon Titan Image 来生成图像。

SDK适用于 Java 2.x

Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用 Amazon Titan 图像生成器创建图片。

```
// 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 JSONObject("/
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);
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [InvokeModel](#)”中的。

亚马逊 Titan 文本

交谈

以下代码示例展示了如何使用 Bedrock 的 Converse 向 Amazon Titan Text 发送短信。API

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用 Bedrock 的 Converse 向 Amazon Titan Text 发送短信。API

```
// Use the Converse API to send a text message to Amazon Titan Text.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Titan Text Premier.
        var modelId = "amazon.titan-text-premier-v1:0";
```

```
        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                    .maxTokens(512)
                    .temperature(0.5F)
                    .topP(0.9F)));

            // Retrieve the generated text from Bedrock's response object.
            var responseText = response.output().message().content().get(0).text();
            System.out.println(responseText);

            return responseText;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
            e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        converse();
    }
}
```

使用 Bedrock 的 Converse 和异步 Java 客户端，向 Amazon Titan Te API 发送短信。

```
// Use the Converse API to send a text message to Amazon Titan Text
```

```
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Titan Text Premier.
        var modelId = "amazon.titan-text-premier-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        // Send the message with a basic inference configuration.
        var request = client.converse(params -> params
            .modelId(modelId)
            .messages(message)
            .inferenceConfig(config -> config
                .maxTokens(512)
                .temperature(0.5F)
                .topP(0.9F))
```

```
);

// Prepare a future object to handle the asynchronous response.
CompletableFuture<String> future = new CompletableFuture<>();

// Handle the response or error using the future object.
request.whenComplete((response, error) -> {
    if (error == null) {
        // Extract the generated text from Bedrock's response object.
        String responseText =
response.output().message().content().get(0).text();
        future.complete(responseText);
    } else {
        future.completeExceptionally(error);
    }
});

try {
    // Wait for the future object to complete and retrieve the generated
text.
    String responseText = future.get();
    System.out.println(responseText);

    return responseText;

} catch (ExecutionException | InterruptedException e) {
    System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
    throw new RuntimeException(e);
}

}

public static void main(String[] args) {
    converseAsync();
}
}
```

- 有关API详细信息，请参阅《AWS SDK for Java 2.x API参考资料》中的“[Converse](#)”。

ConverseStream

以下代码示例展示了如何使用 Bedrock 的 Converse 向 Amazon Titan Text 发送短信API并实时处理响应流。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用 Bedrock 的 Converse 向 Amazon Titan Text 发送短信，API并实时处理响应流。

```
// Use the Converse API to send a text message to Amazon Titan Text
// and print the response stream.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import
    software.amazon.awssdk.services.bedrockruntime.model.ConverseStreamResponseHandler;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.ExecutionException;

public class ConverseStream {

    public static void main(String[] args) {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Titan Text Premier.
        var modelId = "amazon.titan-text-premier-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
```

```
var message = Message.builder()
    .content(ContentBlock.fromText(inputText))
    .role(ConversationRole.USER)
    .build();

// Create a handler to extract and print the response text in real-time.
var responseStreamHandler = ConverseStreamResponseHandler.builder()
    .subscriber(ConverseStreamResponseHandler.Visitor.builder()
        .onContentBlockDelta(chunk -> {
            String responseText = chunk.delta().text();
            System.out.print(responseText);
        }).build())
    .onError(err ->
        System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage())
    ).build();

try {
    // Send the message with a basic inference configuration and attach the
handler.
    client.converseStream(request -> request
        .modelId(modelId)
        .messages(message)
        .inferenceConfig(config -> config
            .maxTokens(512)
            .temperature(0.5F)
            .topP(0.9F)
        ), responseStreamHandler).get();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ConverseStream](#)”中的。

InvokeModel

以下代码示例展示了如何使用调用模型API向 Amazon Titan Text 发送短信。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用调用模型API发送短信。

```
// Use the native inference API to send a text message to Amazon Titan Text.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Titan Text Premier.
        var modelId = "amazon.titan-text-premier-v1:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        titan-text.html
        var nativeRequestTemplate = "{ \"inputText\": \"{{prompt}}\" }";

        // Define the prompt for the model.
```

```
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in the model's native request payload.
String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/results/0/outputText").queryFrom(responseBody).toString();
    System.out.println(text);

    return text;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}


public static void main(String[] args) {
    invokeModel();
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [InvokeModel](#)”中的。

InvokeModelWithResponseStream

以下代码示例演示如何使用调用模型向 Amazon Titan Text 模型API发送短信并打印响应流。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用 Invoke Model API 发送短信并实时处理响应流。

```
// Use the native inference API to send a text message to Amazon Titan Text
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

import java.util.concurrent.ExecutionException;

import static
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
        InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Titan Text Premier.
```

```
var modelId = "amazon.titan-text-premier-v1:0";

// The InvokeModelWithResponseStream API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
titan-text.html
var nativeRequestTemplate = "{ \"inputText\": \"{{prompt}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in the model's native request payload.
String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

// Create a request with the model ID and the model's native request
payload.
var request = InvokeModelWithResponseStreamRequest.builder()
    .body(SdkBytes.fromUtf8String(nativeRequest))
    .modelId(modelId)
    .build();

// Prepare a buffer to accumulate the generated response text.
var completeResponseTextBuffer = new StringBuilder();

// Prepare a handler to extract, accumulate, and print the response text in
real-time.
var responseStreamHandler =
InvokeModelWithResponseStreamResponseHandler.builder()
    .subscriber(Visitor.builder().onChunk(chunk -> {
        // Extract and print the text from the model's native response.
        var response = new JSONObject(chunk.bytes().asUtf8String());
        var text = new JSONPointer("/outputText").queryFrom(response);
        System.out.print(text);

        // Append the text to the response text buffer.
        completeResponseTextBuffer.append(text);
    }).build()).build();

try {
    // Send the request and wait for the handler to process the response.
    client.invokeModelWithResponseStream(request,
responseStreamHandler).get();
```

```
        // Return the complete response text.
        return completeResponseTextBuffer.toString();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
            e.getCause().getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) throws ExecutionException,
    InterruptedException {
    invokeModelWithResponseStream();
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [InvokeModelWithResponseStream](#)”中的。

Amazon Titan Text Embeddings

InvokeModel

以下代码示例展示了如何：

- 开始创建您的第一个嵌入内容。
- 创建嵌入式，配置维度数量和归一化（仅限 V2）。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用 Titan Text Embeddings V2 创建你的第一个嵌入内容。

```
// 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();
}
}

```

调用 Titan Text Embeddings V2，配置维度数量和归一化。

```

/**
 * 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;
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [InvokeModel](#)”中的。

Anthropic Claude

交谈

以下代码示例显示了如何使用 Bedrock 的 Converse 向 Anthropic Claude 发送短信。API

SDK适用于 Java 2.x

 Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用 Bedrock 的 Converse 给 Anthropic Claude 发一条短信。API

```
// Use the Converse API to send a text message to Anthropic Claude.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
```

```
        .build());

    try {
        // Send the message with a basic inference configuration.
        ConverseResponse response = client.converse(request -> request
            .modelId(modelId)
            .messages(message)
            .inferenceConfig(config -> config
                .maxTokens(512)
                .temperature(0.5F)
                .topP(0.9F)));

        // Retrieve the generated text from Bedrock's response object.
        var responseText = response.output().message().content().get(0).text();
        System.out.println(responseText);

        return responseText;
    } catch (SdkClientException e) {
        System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
            e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    converse();
}
}
```

使用 Bedrock 的 Converse 和异步 Java 客户端，向 Anthropic Claude API 发送短信。

```
// Use the Converse API to send a text message to Anthropic Claude
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;
```



```
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        // Send the message with a basic inference configuration.
        var request = client.converse(params -> params
            .modelId(modelId)
            .messages(message)
            .inferenceConfig(config -> config
                .maxTokens(512)
                .temperature(0.5F)
                .topP(0.9F))
        );

        // Prepare a future object to handle the asynchronous response.
        CompletableFuture<String> future = new CompletableFuture<>();

        // Handle the response or error using the future object.
        request.whenComplete((response, error) -> {
            if (error == null) {
```

```
        // Extract the generated text from Bedrock's response object.
        String responseText =
response.output().message().content().get(0).text();
        future.complete(responseText);
    } else {
        future.completeExceptionally(error);
    }
});

try {
    // Wait for the future object to complete and retrieve the generated
text.
    String responseText = future.get();
    System.out.println(responseText);

    return responseText;

} catch (ExecutionException | InterruptedException e) {
    System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
    throw new RuntimeException(e);
}

}

public static void main(String[] args) {
    converseAsync();
}
}
```

- 有关API详细信息，请参阅《AWS SDK for Java 2.x API参考资料》中的“[Converse](#)”。

ConverseStream

以下代码示例显示了如何使用 Bedrock 的 Converse 向 Anthropic Claude 发送短信API并实时处理响应流。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用 Bedrock 的 Converse 向 Anthropic Claude 发送短信，API并实时处理响应流。

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

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ConverseStream](#)”中的。

InvokeModel

以下代码示例显示了如何使用调用模型向 Anthropic Claude 发送短信。API

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用调用模型API发送短信。

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

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [InvokeModel](#)”中的。

InvokeModelWithResponseStream

以下代码示例演示如何使用调用模型向 Anthropic Claude 模型API发送短信并打印响应流。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用 Invoke Model API 发送短信并实时处理响应流。

```
// 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.InvokeModelWithResponseStreamResponse;

import java.util.Objects;
import java.util.concurrent.ExecutionException;

import static
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
        InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
```

```
// Replace the DefaultCredentialsProvider with your preferred credentials
provider.
var client = BedrockRuntimeAsyncClient.builder()
    .credentialsProvider(DefaultCredentialsProvider.create())
    .region(Region.US_EAST_1)
    .build();

// Set the model ID, e.g., Claude 3 Haiku.
var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

// The InvokeModelWithResponseStream API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
anthropic-claude-messages.html
var nativeRequestTemplate = """
    {
        "anthropic_version": "bedrock-2023-05-31",
        "max_tokens": 512,
        "temperature": 0.5,
        "messages": [{
            "role": "user",
            "content": "{{prompt}}"
        }]
    }""";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in the model's native request payload.
String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

// Create a request with the model ID and the model's native request
payload.
var request = InvokeModelWithResponseStreamRequest.builder()
    .body(SdkBytes.fromUtf8String(nativeRequest))
    .modelId(modelId)
    .build();

// Prepare a buffer to accumulate the generated response text.
var completeResponseTextBuffer = new StringBuilder();

// Prepare a handler to extract, accumulate, and print the response text in
real-time.
```



```
    var responseStreamHandler =
    InvokeModelWithResponseStreamResponseHandler.builder()
        .subscriber(Visitor.builder().onChunk(chunk -> {
            var response = new JSONObject(chunk.bytes().asUtf8String());

            // Extract and print the text from the content blocks.
            if (Objects.equals(response.getString("type"),
"content_block_delta")) {
                var text = new JSONPointer("/delta/
text").queryFrom(response);
                System.out.print(text);

                // Append the text to the response text buffer.
                completeResponseTextBuffer.append(text);
            }
        }).build()).build();

    try {
        // Send the request and wait for the handler to process the response.
        client.invokeModelWithResponseStream(request,
responseStreamHandler).get();

        // Return the complete response text.
        return completeResponseTextBuffer.toString();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) throws ExecutionException,
InterruptedException {
    invokeModelWithResponseStream();
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [InvokeModelWithResponseStream](#)”中的。

Cohere Command

交谈

以下代码示例显示了如何使用 Bedrock 的 Converse 向 Cohere Command 发送短信。API

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用 Bedrock 的 Converse 向 Cohere Command 发送短信。API

```
// Use the Converse API to send a text message to Cohere Command.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Command R.
        var modelId = "cohere.command-r-v1:0";
```

```
    // Create the input text and embed it in a message object with the user
    role.
    var inputText = "Describe the purpose of a 'hello world' program in one
    line.";
    var message = Message.builder()
        .content(ContentBlock.fromText(inputText))
        .role(ConversationRole.USER)
        .build();

    try {
        // Send the message with a basic inference configuration.
        ConverseResponse response = client.converse(request -> request
            .modelId(modelId)
            .messages(message)
            .inferenceConfig(config -> config
                .maxTokens(512)
                .temperature(0.5F)
                .topP(0.9F)));

        // Retrieve the generated text from Bedrock's response object.
        var responseText = response.output().message().content().get(0).text();
        System.out.println(responseText);

        return responseText;

    } catch (SdkClientException e) {
        System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
            e.getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) {
    converse();
}
}
```

使用 Bedrock 的 Converse 和异步 Java 客户端，向 Cohere API Command 发送短信。

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

- 有关API详细信息，请参阅《AWS SDK for Java 2.x API参考资料》中的“[Converse](#)”。

ConverseStream

以下代码示例显示了如何使用 Bedrock 的 Converse 向 Cohere Command 发送短信API并实时处理响应流。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用 Bedrock 的 Converse 向 Cohere Command 发送短信，API 然后实时处理响应流。

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

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ConverseStream](#)”中的。

InvokeModel: 命令 R 和 R+

以下代码示例显示了如何使用调用模型向 Cohere Command R 和 R+ 发送短信。API

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用调用模型API发送短信。

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

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [InvokeModel](#)”中的。

InvokeModel: 命令和命令灯

以下代码示例显示了如何使用调用模型API向 Cohere Command 发送短信。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用调用模型API发送短信。

```
// Use the native inference API to send a text message to Cohere Command.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class Command_InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Command Light.
        var modelId = "cohere.command-light-text-v14";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        cohere-command.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{prompt}}\" }";

        // Define the prompt for the model.
```

```
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in the model's native request payload.
String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/generations/0/text").queryFrom(responseBody).toString();
    System.out.println(text);

    return text;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}

public static void main(String[] args) {
    invokeModel();
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [InvokeModel](#)”中的。

InvokeModelWithResponseStream: 命令 R 和 R+

以下代码示例演示如何使用API带有响应流的调用模型向 Cohere Command 发送短信。

SDK适用于 Java 2.x

 Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用 Invoke Model API 发送短信并实时处理响应流。

```
// 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.InvokeModelWithResponseStreamResponse;

import java.util.concurrent.ExecutionException;

import static
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class Command_R_InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
        InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Command R.
```

```
var modelId = "cohere.command-r-v1:0";

// The InvokeModelWithResponseStream API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
cohere-command-r-plus.html
var nativeRequestTemplate = "{ \"message\": \"{{prompt}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in the model's native request payload.
String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

// Create a request with the model ID and the model's native request
payload.
var request = InvokeModelWithResponseStreamRequest.builder()
    .body(SdkBytes.fromUtf8String(nativeRequest))
    .modelId(modelId)
    .build();

// Prepare a buffer to accumulate the generated response text.
var completeResponseTextBuffer = new StringBuilder();

// Prepare a handler to extract, accumulate, and print the response text in
real-time.
var responseStreamHandler =
InvokeModelWithResponseStreamResponseHandler.builder()
    .subscriber(Visitor.builder().onChunk(chunk -> {
        // Extract and print the text from the model's native response.
        var response = new JSONObject(chunk.bytes().asUtf8String());
        var text = new JSONPointer("/text").queryFrom(response);
        System.out.print(text);

        // Append the text to the response text buffer.
        completeResponseTextBuffer.append(text);
    }).build()).build();

try {
    // Send the request and wait for the handler to process the response.
    client.invokeModelWithResponseStream(request,
responseStreamHandler).get();
```

```

        // Return the complete response text.
        return completeResponseTextBuffer.toString();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
            e.getCause().getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) throws ExecutionException,
    InterruptedException {
    invokeModelWithResponseStream();
}
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [InvokeModel](#)”中的。

InvokeModelWithResponseStream: 命令和命令灯

以下代码示例演示如何使用API带有响应流的调用模型向 Cohere Command 发送短信。

SDK适用于 Java 2.x

Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用 Invoke Model API 发送短信并实时处理响应流。

```

// Use the native inference API to send a text message to Cohere Command
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;

```

```
import
software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

import java.util.concurrent.ExecutionException;

import static
software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class Command_InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
    InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Command Light.
        var modelId = "cohere.command-light-text-v14";

        // The InvokeModelWithResponseStream API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        cohere-command.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{prompt}}\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in the model's native request payload.
        String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

        // Create a request with the model ID and the model's native request
        payload.
        var request = InvokeModelWithResponseStreamRequest.builder()
            .body(SdkBytes.fromUtf8String(nativeRequest))
            .modelId(modelId)
```

```
        .build();

        // Prepare a buffer to accumulate the generated response text.
        var completeResponseTextBuffer = new StringBuilder();

        // Prepare a handler to extract, accumulate, and print the response text in
        real-time.
        var responseStreamHandler =
        InvokeModelWithResponseStreamResponseHandler.builder()
            .subscriber(Visitor.builder().onChunk(chunk -> {
                // Extract and print the text from the model's native response.
                var response = new JSONObject(chunk.bytes().asUtf8String());
                var text = new JSONPointer("/generations/0/
text").queryFrom(response);
                System.out.print(text);

                // Append the text to the response text buffer.
                completeResponseTextBuffer.append(text);
            }).build()).build();

        try {
            // Send the request and wait for the handler to process the response.
            client.invokeModelWithResponseStream(request,
            responseStreamHandler).get();

            // Return the complete response text.
            return completeResponseTextBuffer.toString();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
            e.getCause().getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) throws ExecutionException,
    InterruptedException {
        invokeModelWithResponseStream();
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [InvokeModel](#)”中的。

Meta Llama

交谈

以下代码示例展示了如何使用 Bedrock 的 Converse 向 Meta Llama 发送短信。API

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用 Bedrock 的 Converse 向 Meta Llama 发送短信。API

```
// Use the Converse API to send a text message to Meta Llama.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";
```

```
        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                    .maxTokens(512)
                    .temperature(0.5F)
                    .topP(0.9F)));

            // Retrieve the generated text from Bedrock's response object.
            var responseText = response.output().message().content().get(0).text();
            System.out.println(responseText);

            return responseText;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
            e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        converse();
    }
}
```

使用 Bedrock 的 Converse API 和异步 Java 客户端，向 Meta Llama 发送短信。

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

- 有关API详细信息，请参阅《AWS SDK for Java 2.x API参考资料》中的“[Converse](#)”。

ConverseStream

以下代码示例展示了如何使用 Bedrock 的 Converse 向 Meta Llama 发送短信API并实时处理响应流。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用 Bedrock 的 Converse 向 Meta Llama 发送短信API并实时处理响应流。

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

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ConverseStream](#)”中的。

InvokeModel: Llama 2

以下代码示例展示了如何使用调用模型API向 Meta Llama 2 发送短信。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用调用模型API发送短信。

```
// Use the native inference API to send a text message to Meta Llama 2.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class Llama2_InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Llama 2 Chat 13B.
        var modelId = "meta.llama2-13b-chat-v1";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        meta.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

        // Define the prompt for the model.
```

```
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in Llama 2's instruction format.
var instruction = "<s>[INST] {{prompt}} [/INST]\\n".replace("{{prompt}}",
prompt);

// Embed the instruction in the the native request payload.
var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
instruction);

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/
generation").queryFrom(responseBody).toString();
    System.out.println(text);

    return text;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}

}

public static void main(String[] args) {
    invokeModel();
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [InvokeModel](#)”中的。

InvokeModel: Llama 3

以下代码示例展示了如何使用调用模型API向 Meta Llama 3 发送短信。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用调用模型API发送短信。

```
// Use the native inference API to send a text message to Meta Llama 3.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class Llama3_InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
        var modelId = "meta.llama3-8b-instruct-v1:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
```

```
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
meta.html
var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in Llama 3's instruction format.
var instruction = (
    "<|begin_of_text|>\n" +
    "<|start_header_id|>user<|end_header_id|>\n" +
    "{{prompt}} <|eot_id|>\n" +
    "<|start_header_id|>assistant<|end_header_id|>\n"
).replace("{{prompt}}", prompt);

// Embed the instruction in the the native request payload.
var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
instruction);

try {
    // Encode and send the request to the Bedrock Runtime.
    var response = client.invokeModel(request -> request
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
    );

    // Decode the response body.
    var responseBody = new JSONObject(response.body().asUtf8String());

    // Retrieve the generated text from the model's response.
    var text = new JSONPointer("/
generation").queryFrom(responseBody).toString();
    System.out.println(text);

    return text;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}
}

public static void main(String[] args) {
```

```
        invokeModel();
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [InvokeModel](#)”中的。

InvokeModelWithResponseStream: Llama 2

以下代码示例演示如何使用调用模型API向 Meta Llama 2 发送短信并打印响应流。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用 Invoke Model API 发送短信并实时处理响应流。

```
// Use the native inference API to send a text message to Meta Llama 2
// and print the response stream.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamRequest;
import
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

import java.util.concurrent.ExecutionException;

import static
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class Llama2_InvokeModelWithResponseStream {
```

```
public static String invokeModelWithResponseStream() throws ExecutionException,
InterruptedException {

    // Create a Bedrock Runtime client in the AWS Region you want to use.
    // Replace the DefaultCredentialsProvider with your preferred credentials
    provider.
    var client = BedrockRuntimeAsyncClient.builder()
        .credentialsProvider(DefaultCredentialsProvider.create())
        .region(Region.US_EAST_1)
        .build();

    // Set the model ID, e.g., Llama 2 Chat 13B.
    var modelId = "meta.llama2-13b-chat-v1";

    // The InvokeModelWithResponseStream API uses the model's native payload.
    // Learn more about the available inference parameters and response fields
    at:
    // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
    meta.html
    var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

    // Define the prompt for the model.
    var prompt = "Describe the purpose of a 'hello world' program in one line.";

    // Embed the prompt in Llama 2's instruction format.
    var instruction = "<s>[INST] {{prompt}} [/INST]\\n".replace("{{prompt}}",
    prompt);

    // Embed the instruction in the the native request payload.
    var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
    instruction);

    // Create a request with the model ID and the model's native request
    payload.
    var request = InvokeModelWithResponseStreamRequest.builder()
        .body(SdkBytes.fromUtf8String(nativeRequest))
        .modelId(modelId)
        .build();

    // Prepare a buffer to accumulate the generated response text.
    var completeResponseTextBuffer = new StringBuilder();

    // Prepare a handler to extract, accumulate, and print the response text in
    real-time.
```

```
var responseStreamHandler =
InvokeModelWithResponseStreamResponseHandler.builder()
    .subscriber(Visitor.builder().onChunk(chunk -> {
        // Extract and print the text from the model's native response.
        var response = new JSONObject(chunk.bytes().asUtf8String());
        var text = new JSONPointer("/generation").queryFrom(response);
        System.out.print(text);

        // Append the text to the response text buffer.
        completeResponseTextBuffer.append(text);
    }).build()).build();

try {
    // Send the request and wait for the handler to process the response.
    client.invokeModelWithResponseStream(request,
responseStreamHandler).get();

    // Return the complete response text.
    return completeResponseTextBuffer.toString();

} catch (ExecutionException | InterruptedException e) {
    System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
    throw new RuntimeException(e);
}

}


public static void main(String[] args) throws ExecutionException,
InterruptedException {
    invokeModelWithResponseStream();
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [InvokeModelWithResponseStream](#)”中的。

InvokeModelWithResponseStream: Llama 3

以下代码示例展示了如何使用调用模型API向 Meta Llama 3 发送短信并打印响应流。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用 Invoke Model API 发送短信并实时处理响应流。

```
// 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.InvokeModelWithResponseStreamResponse;

import java.util.concurrent.ExecutionException;

import static
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class Llama3_InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
        InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Llama 3 8b Instruct.
```

```
var modelId = "meta.llama3-8b-instruct-v1:0";

// The InvokeModelWithResponseStream API uses the model's native payload.
// Learn more about the available inference parameters and response fields
at:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
meta.html
var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

// Define the prompt for the model.
var prompt = "Describe the purpose of a 'hello world' program in one line.";

// Embed the prompt in Llama 3's instruction format.
var instruction = (
    "<|begin_of_text|>\n" +
    "<|start_header_id|>user<|end_header_id|>\n" +
    "{{prompt}} <|eot_id|>\n" +
    "<|start_header_id|>assistant<|end_header_id|>\n"
).replace("{{prompt}}", prompt);

// Embed the instruction in the the native request payload.
var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
instruction);

// Create a request with the model ID and the model's native request
payload.
var request = InvokeModelWithResponseStreamRequest.builder()
    .body(SdkBytes.fromUtf8String(nativeRequest))
    .modelId(modelId)
    .build();

// Prepare a buffer to accumulate the generated response text.
var completeResponseTextBuffer = new StringBuilder();

// Prepare a handler to extract, accumulate, and print the response text in
real-time.
var responseStreamHandler =
InvokeModelWithResponseStreamResponseHandler.builder()
    .subscriber(Visitor.builder().onChunk(chunk -> {
        // Extract and print the text from the model's native response.
        var response = new JSONObject(chunk.bytes().asUtf8String());
        var text = new JSONPointer("/generation").queryFrom(response);
        System.out.print(text);
```

```
        // Append the text to the response text buffer.
        completeResponseTextBuffer.append(text);
    }).build()).build();

    try {
        // Send the request and wait for the handler to process the response.
        client.invokeModelWithResponseStream(request,
responseStreamHandler).get();

        // Return the complete response text.
        return completeResponseTextBuffer.toString();

    } catch (ExecutionException | InterruptedException e) {
        System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
        throw new RuntimeException(e);
    }
}

public static void main(String[] args) throws ExecutionException,
InterruptedException {
    invokeModelWithResponseStream();
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [InvokeModelWithResponseStream](#)”中的。

Mistral AI

交谈

以下代码示例显示了如何使用 Bedrock 的 Converse 向 Mistral 发送短信。API

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用 Bedrock 的 Converse 向 Mistral 发送短信。API

```
// Use the Converse API to send a text message to Mistral.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // Create the input text and embed it in a message object with the user
        role.
        var inputText = "Describe the purpose of a 'hello world' program in one
        line.";
        var message = Message.builder()
            .content(ContentBlock.fromText(inputText))
            .role(ConversationRole.USER)
            .build();

        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
```

```
        .maxTokens(512)
        .temperature(0.5F)
        .topP(0.9F));

    // Retrieve the generated text from Bedrock's response object.
    var responseText = response.output().message().content().get(0).text();
    System.out.println(responseText);

    return responseText;

} catch (SdkClientException e) {
    System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
    throw new RuntimeException(e);
}

}

public static void main(String[] args) {
    converse();
}
}
```

使用 Bedrock 的 Converse API 和异步 Java 客户端，向 Mistral 发送短信。

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

- 有关API详细信息，请参阅《AWS SDK for Java 2.x API参考资料》中的“[Converse](#)”。

ConverseStream

以下代码示例显示了如何使用 Bedrock 的 Converse 向 Mistral 发送短信API并实时处理响应流。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用 Bedrock 的 Converse 向 Mistral 发送短信API并实时处理响应流。

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

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ConverseStream](#)”中的。

InvokeModel

以下代码示例显示了如何使用调用模型向 Mistral 模型发送短信。API

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用调用模型API发送短信。

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

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [InvokeModel](#)”中的。

InvokeModelWithResponseStream

以下代码示例演示如何使用调用模型向 Mistral AI 模型API发送短信并打印响应流。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用 Invoke Model API 发送短信并实时处理响应流。

```
// 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.InvokeModelWithResponseStreamResponse;

import java.util.concurrent.ExecutionException;

import static
    software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponse;

public class InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() throws ExecutionException,
        InterruptedException {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
        provider.
        var client = BedrockRuntimeAsyncClient.builder()
            .credentialsProvider(DefaultCredentialsProvider.create())
            .region(Region.US_EAST_1)
            .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // The InvokeModelWithResponseStream API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
        at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
        mistral-text-completion.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in Mistral's instruction format.
```

```
        var instruction = "<s>[INST] {{prompt}} [/INST]\\n".replace("{{prompt}}",
prompt);

        // Embed the instruction in the the native request payload.
        var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
instruction);

        // Create a request with the model ID and the model's native request
payload.
        var request = InvokeModelWithResponseStreamRequest.builder()
            .body(SdkBytes.fromUtf8String(nativeRequest))
            .modelId(modelId)
            .build();

        // Prepare a buffer to accumulate the generated response text.
        var completeResponseTextBuffer = new StringBuilder();

        // Prepare a handler to extract, accumulate, and print the response text in
real-time.
        var responseStreamHandler =
InvokeModelWithResponseStreamResponseHandler.builder()
            .subscriber(Visitor.builder().onChunk(chunk -> {
                // Extract and print the text from the model's native response.
                var response = new JSONObject(chunk.bytes().asUtf8String());
                var text = new JSONPointer("/outputs/0/
text").queryFrom(response);
                System.out.print(text);

                // Append the text to the response text buffer.
                completeResponseTextBuffer.append(text);
            }).build()).build();

        try {
            // Send the request and wait for the handler to process the response.
            client.invokeModelWithResponseStream(request,
responseStreamHandler).get();

            // Return the complete response text.
            return completeResponseTextBuffer.toString();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
e.getCause().getMessage());
            throw new RuntimeException(e);
        }
    }
}
```

```
    }  
  }  
  
  public static void main(String[] args) throws ExecutionException,  
    InterruptedException {  
    invokeModelWithResponseStream();  
  }  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [InvokeModelWithResponseStream](#)”中的。

场景

创建用于与 Amazon Bedrock 基础模型进行交互的平台应用程序

以下代码示例展示了如何创建平台，通过不同的模式与 Amazon Bedrock 基础模型进行交互。

SDK适用于 Java 2.x

Java Foundation Model (FM) Playground 是一款 Spring Boot 示例应用程序，演示了如何将 Amazon Bedrock 与 Java 结合使用。此示例演示 Java 开发人员可如何使用 Amazon Bedrock 来构建支持生成式人工智能的应用程序。您可以使用以下三个操场测试 Amazon Bedrock 基础模型并与之交互：

- 文本操场。
- 聊天操场。
- 图像操场。

该示例还列出并显示您可以访问的基础模型及其特点。有关源代码和部署说明，请参阅中的项目 [GitHub](#)。

本示例中使用的服务

- Amazon Bedrock 运行时系统

Stable Diffusion

InvokeModel

以下代码示例展示了如何在 Amazon Bedrock 上调用 Stability.ai Stable Diffusion XL 来生成图像。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用“稳定扩散”创建图像。

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

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [InvokeModel](#)”中的。

CloudFront 用SDK于 Java 2.x 的示例

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 CloudFront。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)
- [场景](#)

操作

CreateDistribution

以下代码示例演示如何使用 CreateDistribution。

SDK适用于 Java 2.x

 Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

以下示例使用 Amazon Simple Storage Service (Amazon S3) 桶作为内容来源。

创建发行版后，代码会创建一个 [CloudFrontWaiter](#)，等待发行版部署完毕后再返回发行版。

```
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))

> 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;

```

```
    }  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateDistribution](#)”中的。

CreateFunction

以下代码示例演示如何使用 CreateFunction。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateFunction](#)”中的。

CreateKeyGroup

以下代码示例演示如何使用 CreateKeyGroup。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

密钥组需要至少一个用于验证已签名URLs或 Cookie 的公钥。

```
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;
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateKeyGroup](#)”中的。

CreatePublicKey

以下代码示例演示如何使用 CreatePublicKey。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

以下代码示例读取公钥并将其上传到 Amazon CloudFront。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.CreatePublicKeyResponse;
import software.amazon.awssdk.utils.IoUtils;

import java.io.IOException;
import java.io.InputStream;
import java.util.UUID;

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

    public static String createPublicKey(CloudFrontClient cloudFrontClient, String
        publicKeyFileName) {
        try (InputStream is =
            CreatePublicKey.class.getClassLoader().getResourceAsStream(publicKeyFileName)) {
```

```
String publicKeyString = IoUtils.toUtf8String(is);
CreatePublicKeyResponse createPublicKeyResponse = cloudFrontClient
    .createPublicKey(b -> b.publicKeyConfig(c -> c
        .name("JavaCreatedPublicKey" + UUID.randomUUID())
        .encodedKey(publicKeyString)
        .callerReference(UUID.randomUUID().toString())));
String createdPublicKeyId = createPublicKeyResponse.publicKey().id();
logger.info("Public key created with id: [{}]", createdPublicKeyId);
return createdPublicKeyId;

    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreatePublicKey](#)”中的。

DeleteDistribution

以下代码示例演示如何使用 DeleteDistribution。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

以下代码示例将分配更新为禁用，使用 waiter 等待更改部署完成，然后删除该分配。

```
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);
        }
    }
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteDistribution](#)”中的。

UpdateDistribution

以下代码示例演示如何使用 UpdateDistribution。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [UpdateDistribution](#)”中的。

场景

删除签名资源

以下代码示例演示了如何删除用于访问 Amazon Simple Storage Service (Amazon S3) 桶中的受限内容的资源。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [DeleteKeyGroup](#)
 - [DeleteOriginAccessControl](#)
 - [DeletePublicKey](#)

签名URLs和饼干

以下代码示例显示了如何创建允许访问受限资源的签名URLs和 Cookie。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用[CannedSignerRequest](#)课堂签名URLs或使用罐装政策制作饼干。

```
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();
}
}
```

使用[CustomSignerRequest](#)课堂签名URLs或使用自定义策略进行 Cookie。activeDate 和 ipRange 是可选方法。

```
import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;

import java.net.URL;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.temporal.ChronoUnit;

public class CreateCustomPolicyRequest {

    public static CustomSignerRequest createRequestForCustomPolicy(String
distributionDomainName,
        String fileNameToUpload,
        String privateKeyFullPath, String publicKeyId) throws Exception {
        String protocol = "https";
        String resourcePath = "/" + fileNameToUpload;
```

```

        String cloudFrontUrl = new URL(protocol, distributionDomainName,
resourcePath).toString();
        Instant expireDate = Instant.now().plus(7, ChronoUnit.DAYS);
        // URL will be accessible tomorrow using the signed URL.
        Instant activeDate = Instant.now().plus(1, ChronoUnit.DAYS);
        Path path = Paths.get(privateKeyFullPath);

        return CustomSignerRequest.builder()
            .resourceUrl(cloudFrontUrl)
            .privateKey(path)
            .keyPairId(publicKeyId)
            .expirationDate(expireDate)
            .activeDate(activeDate) // Optional.
            // .ipRange("192.168.0.1/24") // Optional.
            .build();
    }
}

```

以下示例演示如何使用该[CloudFrontUtilities](#)类生成签名 Cookie 和URLs。在上@@ [查看](#)此代码示例 GitHub。

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudfront.CloudFrontUtilities;
import software.amazon.awssdk.services.cloudfront.cookie.CookiesForCannedPolicy;
import software.amazon.awssdk.services.cloudfront.cookie.CookiesForCustomPolicy;
import software.amazon.awssdk.services.cloudfront.model.CannedSignerRequest;
import software.amazon.awssdk.services.cloudfront.model.CustomSignerRequest;
import software.amazon.awssdk.services.cloudfront.url.SignedUrl;

public class SigningUtilities {
    private static final Logger logger =
LoggerFactory.getLogger(SigningUtilities.class);
    private static final CloudFrontUtilities cloudFrontUtilities =
CloudFrontUtilities.create();

    public static SignedUrl signUrlForCannedPolicy(CannedSignerRequest
cannedSignerRequest) {
        SignedUrl signedUrl =
cloudFrontUtilities.getSignedUrlWithCannedPolicy(cannedSignerRequest);
        logger.info("Signed URL: [{}]", signedUrl.url());
        return signedUrl;
    }
}

```



```
    }

    public static SignedUrl signUrlForCustomPolicy(CustomSignerRequest
customSignerRequest) {
        SignedUrl signedUrl =
cloudFrontUtilities.getSignedUrlWithCustomPolicy(customSignerRequest);
        logger.info("Signed URL: [{}]", signedUrl.url());
        return signedUrl;
    }

    public static CookiesForCannedPolicy
getCookiesForCannedPolicy(CannedSignerRequest cannedSignerRequest) {
        CookiesForCannedPolicy cookiesForCannedPolicy = cloudFrontUtilities
                .getCookiesForCannedPolicy(cannedSignerRequest);
        logger.info("Cookie EXPIRES header [{}]",
cookiesForCannedPolicy.expiresHeaderValue());
        logger.info("Cookie KEYPAIR header [{}]",
cookiesForCannedPolicy.keyPairIdHeaderValue());
        logger.info("Cookie SIGNATURE header [{}]",
cookiesForCannedPolicy.signatureHeaderValue());
        return cookiesForCannedPolicy;
    }

    public static CookiesForCustomPolicy
getCookiesForCustomPolicy(CustomSignerRequest customSignerRequest) {
        CookiesForCustomPolicy cookiesForCustomPolicy = cloudFrontUtilities
                .getCookiesForCustomPolicy(customSignerRequest);
        logger.info("Cookie POLICY header [{}]",
cookiesForCustomPolicy.policyHeaderValue());
        logger.info("Cookie KEYPAIR header [{}]",
cookiesForCustomPolicy.keyPairIdHeaderValue());
        logger.info("Cookie SIGNATURE header [{}]",
cookiesForCustomPolicy.signatureHeaderValue());
        return cookiesForCustomPolicy;
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CloudFrontUtilities](#)”中的。

CloudWatch 用SDK于 Java 2.x 的示例

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 CloudWatch。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

开始使用

你好 CloudWatch

以下代码示例展示了如何开始使用 CloudWatch。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListMetrics](#)”中的。

主题

- [操作](#)
- [场景](#)

操作

DeleteAlarms

以下代码示例演示如何使用 DeleteAlarms。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteAlarm {
    public static void main(String[] args) {
        final String usage = ""
```

```
Usage:
    <alarmName>

Where:
    alarmName - An alarm name to delete (for example, MyAlarm).
    """;

if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

String alarmName = args[0];
Region region = Region.US_EAST_2;
CloudWatchClient cw = CloudWatchClient.builder()
    .region(region)
    .build();

deleteCWAlarm(cw, alarmName);
cw.close();
}

public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {
    try {
        DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
            .alarmNames(alarmName)
            .build();

        cw.deleteAlarms(request);
        System.out.printf("Successfully deleted alarm %s", alarmName);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteAlarms](#)”中的。

DeleteAnomalyDetector

以下代码示例演示如何使用 DeleteAnomalyDetector。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void deleteAnomalyDetector(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .stat("Maximum")
            .build();

        DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
            .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
            .build();

        cw.deleteAnomalyDetector(request);
        System.out.println("Successfully deleted the Anomaly Detector.");

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

```
    }  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteAnomalyDetector](#)”中的。

DeleteDashboards

以下代码示例演示如何使用 DeleteDashboards。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void deleteDashboard(CloudWatchClient cw, String dashboardName) {  
    try {  
        DeleteDashboardsRequest dashboardsRequest =  
DeleteDashboardsRequest.builder()  
            .dashboardNames(dashboardName)  
            .build();  
        cw.deleteDashboards(dashboardsRequest);  
        System.out.println(dashboardName + " was successfully deleted.");  
    } catch (CloudWatchException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteDashboards](#)”中的。

DescribeAlarmHistory

以下代码示例演示如何使用 DescribeAlarmHistory。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void getAlarmHistory(CloudWatchClient cw, String fileName, String
date) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String alarmName = rootNode.findValue("exampleAlarmName").asText();

        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();
        DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
            .startDate(start)
            .endDate(endDate)
            .alarmName(alarmName)
            .historyItemType(HistoryItemType.ACTION)
            .build();

        DescribeAlarmHistoryResponse response =
cw.describeAlarmHistory(historyRequest);
        List<AlarmHistoryItem> historyItems = response.alarmHistoryItems();
        if (historyItems.isEmpty()) {
            System.out.println("No alarm history data found for " + alarmName +
".");
        } else {
            for (AlarmHistoryItem item : historyItems) {
                System.out.println("History summary: " + item.historySummary());
                System.out.println("Time stamp: " + item.timestamp());
            }
        }
    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
    }
}
```



```
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeAlarmHistory](#)”中的。

DescribeAlarms

以下代码示例演示如何使用 DescribeAlarms。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void describeAlarms(CloudWatchClient cw) {
    try {
        List<AlarmType> typeList = new ArrayList<>();
        typeList.add(AlarmType.METRIC_ALARM);

        DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
            .alarmTypes(typeList)
            .maxRecords(10)
            .build();

        DescribeAlarmsResponse response = cw.describeAlarms(alarmsRequest);
        List<MetricAlarm> alarmList = response.metricAlarms();
        for (MetricAlarm alarm : alarmList) {
            System.out.println("Alarm name: " + alarm.alarmName());
            System.out.println("Alarm description: " +
alarm.alarmDescription());
        }
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeAlarms](#)”中的。

DescribeAlarmsForMetric

以下代码示例演示如何使用 DescribeAlarmsForMetric。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void checkForMetricAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        boolean hasAlarm = false;
        int retries = 10;

        DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        while (!hasAlarm && retries > 0) {
            DescribeAlarmsForMetricResponse response =
cw.describeAlarmsForMetric(metricRequest);
            hasAlarm = response.hasMetricAlarms();
            retries--;
            Thread.sleep(20000);
            System.out.println(".");
        }
    }
}
```

```
    }
    if (!hasAlarm)
        System.out.println("No Alarm state found for " + customMetricName +
" after 10 retries.");
    else
        System.out.println("Alarm state found for " + customMetricName +
".");
} catch (CloudWatchException | IOException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeAlarmsForMetric](#)”中的。

DescribeAnomalyDetectors

以下代码示例演示如何使用 DescribeAnomalyDetectors。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void describeAnomalyDetectors(CloudWatchClient cw, String
fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
```

```
        .maxResults(10)
        .metricName(customMetricName)
        .namespace(customMetricNamespace)
        .build();

    DescribeAnomalyDetectorsResponse response =
    cw.describeAnomalyDetectors(detectorsRequest);
    List<AnomalyDetector> anomalyDetectorList = response.anomalyDetectors();
    for (AnomalyDetector detector : anomalyDetectorList) {
        System.out.println("Metric name: " +
    detector.singleMetricAnomalyDetector().metricName());
        System.out.println("State: " + detector.stateValue());
    }

} catch (CloudWatchException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeAnomalyDetectors](#)”中的。

DisableAlarmActions

以下代码示例演示如何使用 `DisableAlarmActions`。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DisableAlarmActions](#)”中的。

EnableAlarmActions

以下代码示例演示如何使用 EnableAlarmActions。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [EnableAlarmActions](#)”中的。

GetMetricData

以下代码示例演示如何使用 GetMetricData。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void getCustomMetricData(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        // Set the date.
        Instant nowDate = Instant.now();

        long hours = 1;
        long minutes = 30;
        Instant date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(minutes,
ChronoUnit.MINUTES);

        Metric met = Metric.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        MetricStat metStat = MetricStat.builder()
            .stat("Maximum")
            .period(1)
            .metric(met)
            .build();

        MetricDataQuery dataQuery = MetricDataQuery.builder()
            .metricStat(metStat)
            .id("foo2")
            .returnData(true)
```



```
        .build();

List<MetricDataQuery> dq = new ArrayList<>();
dq.add(dataQuery);

GetMetricDataRequest getMetReq = GetMetricDataRequest.builder()
    .maxDatapoints(10)
    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
    .startTime(nowDate)
    .endTime(date2)
    .metricDataQueries(dq)
    .build();

GetMetricDataResponse response = cw.getMetricData(getMetReq);
List<MetricDataResult> data = response.metricDataResults();
for (MetricDataResult item : data) {
    System.out.println("The label is " + item.label());
    System.out.println("The status code is " +
item.statusCode().toString());
}

} catch (CloudWatchException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetMetricData](#)”中的。

GetMetricStatistics

以下代码示例演示如何使用 `GetMetricStatistics`。

SDK适用于 Java 2.x

Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void getAndDisplayMetricStatistics(CloudWatchClient cw, String
nameSpace, String metVal,
        String metricOption, String date, Dimension myDimension) {
    try {
        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();

        GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
        .endTime(endDate)
        .startTime(start)
        .dimensions(myDimension)
        .metricName(metVal)
        .namespace(nameSpace)
        .period(86400)
        .statistics(Statistic.fromValue(metricOption))
        .build();

        GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
        List<Datapoint> data = response.datapoints();
        if (!data.isEmpty()) {
            for (Datapoint datapoint : data) {
                System.out
                    .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
            }
        } else {
            System.out.println("The returned data list is empty");
        }

    } catch (CloudWatchException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetMetricStatistics](#)”中的。

GetMetricWidgetImage

以下代码示例演示如何使用 `GetMetricWidgetImage`。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void getAndOpenMetricImage(CloudWatchClient cw, String fileName) {
    System.out.println("Getting Image data for custom metric.");
    try {
        String myJSON = "{\n" +
            "  \"title\": \"Example Metric Graph\",\n" +
            "  \"view\": \"timeSeries\",\n" +
            "  \"stacked\": false,\n" +
            "  \"period\": 10,\n" +
            "  \"width\": 1400,\n" +
            "  \"height\": 600,\n" +
            "  \"metrics\": [\n" +
            "    [\n" +
            "      \"AWS/Billing\",\n" +
            "      \"EstimatedCharges\",\n" +
            "      \"Currency\",\n" +
            "      \"USD\"\n" +
            "    ]\n" +
            "  ]\n" +
            "}";

        GetMetricWidgetImageRequest imageRequest =
            GetMetricWidgetImageRequest.builder()
                .metricWidget(myJSON)
                .build();

        GetMetricWidgetImageResponse response =
            cw.getMetricWidgetImage(imageRequest);
        SdkBytes sdkBytes = response.metricWidgetImage();
        byte[] bytes = sdkBytes.asByteArray();
        File outputFile = new File(fileName);
        try (FileOutputStream outputStream = new FileOutputStream(outputFile)) {
```

```
        outputStream.write(bytes);
    }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetMetricWidgetImage](#)”中的。

ListDashboards

以下代码示例演示如何使用 ListDashboards。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void listDashboards(CloudWatchClient cw) {
    try {
        ListDashboardsIterable listRes = cw.listDashboardsPaginator();
        listRes.stream()
            .flatMap(r -> r.dashboardEntries().stream())
            .forEach(entry -> {
                System.out.println("Dashboard name is: " +
entry.dashboardName());
                System.out.println("Dashboard ARN is: " +
entry.dashboardArn());
            });

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListDashboards](#)”中的。

ListMetrics

以下代码示例演示如何使用 ListMetrics。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Metric;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListMetrics {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <namespace>\s

                Where:
                namespace - The namespace to filter against (for example, AWS/
                EC2).\s

                """;
```

```
    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String namespace = args[0];
    Region region = Region.US_EAST_1;
    CloudWatchClient cw = CloudWatchClient.builder()
        .region(region)
        .build();

    listMets(cw, namespace);
    cw.close();
}

public static void listMets(CloudWatchClient cw, String namespace) {
    boolean done = false;
    String nextToken = null;

    try {
        while (!done) {

            ListMetricsResponse response;
            if (nextToken == null) {
                ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .build();

                response = cw.listMetrics(request);
            } else {
                ListMetricsRequest request = ListMetricsRequest.builder()
                    .namespace(namespace)
                    .nextToken(nextToken)
                    .build();

                response = cw.listMetrics(request);
            }

            for (Metric metric : response.metrics()) {
                System.out.printf("Retrieved metric %s", metric.metricName());
                System.out.println();
            }
        }
    }
}
```

```
        if (response.nextToken() == null) {
            done = true;
        } else {
            nextToken = response.nextToken();
        }
    }

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListMetrics](#)”中的。

PutAnomalyDetector

以下代码示例演示如何使用 PutAnomalyDetector。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void addAnomalyDetector(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
```

```
        .metricName(customMetricName)
        .namespace(customMetricNamespace)
        .stat("Maximum")
        .build();

    PutAnomalyDetectorRequest anomalyDetectorRequest =
PutAnomalyDetectorRequest.builder()
        .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
        .build();

    cw.putAnomalyDetector(anomalyDetectorRequest);
    System.out.println("Added anomaly detector for metric " +
customMetricName + ".");

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [PutAnomalyDetector](#)”中的。

PutDashboard

以下代码示例演示如何使用 PutDashboard。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void createDashboardWithMetrics(CloudWatchClient cw, String
dashboardName, String fileName) {
    try {
        PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
            .dashboardName(dashboardName)
            .dashboardBody(readFileAsString(fileName))
```



```
        .build();

        PutDashboardResponse response = cw.putDashboard(dashboardRequest);
        System.out.println(dashboardName + " was successfully created.");
        List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
        if (messages.isEmpty()) {
            System.out.println("There are no messages in the new Dashboard");
        } else {
            for (DashboardValidationMessage message : messages) {
                System.out.println("Message is: " + message.message());
            }
        }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [PutDashboard](#)”中的。

PutMetricAlarm

以下代码示例演示如何使用 PutMetricAlarm。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static String createAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
```

```
String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
String customMetricName =
rootNode.findValue("customMetricName").asText();
String alarmName = rootNode.findValue("exampleAlarmName").asText();
String emailTopic = rootNode.findValue("emailTopic").asText();
String accountId = rootNode.findValue("accountId").asText();
String region = rootNode.findValue("region").asText();

// Create a List for alarm actions.
List<String> alarmActions = new ArrayList<>();
alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);
PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
    .alarmActions(alarmActions)
    .alarmDescription("Example metric alarm")
    .alarmName(alarmName)

.comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
    .threshold(100.00)
    .metricName(customMetricName)
    .namespace(customMetricNamespace)
    .evaluationPeriods(1)
    .period(10)
    .statistic("Maximum")
    .datapointsToAlarm(1)
    .treatMissingData("ignore")
    .build();

cw.putMetricAlarm(alarmRequest);
System.out.println(alarmName + " was successfully created!");
return alarmName;

} catch (CloudWatchException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [PutMetricAlarm](#)”中的。

PutMetricData

以下代码示例演示如何使用 PutMetricData。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void addMetricDataForAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        // Set an Instant object.
        String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
        Instant instant = Instant.parse(time);

        MetricDatum datum = MetricDatum.builder()
            .metricName(customMetricName)
            .unit(StandardUnit.NONE)
            .value(1001.00)
            .timestamp(instant)
            .build();

        MetricDatum datum2 = MetricDatum.builder()
            .metricName(customMetricName)
            .unit(StandardUnit.NONE)
            .value(1002.00)
            .timestamp(instant)
            .build();

        List<MetricDatum> metricDataList = new ArrayList<>();
```

```
metricDataList.add(datum);
metricDataList.add(datum2);

PutMetricDataRequest request = PutMetricDataRequest.builder()
    .namespace(customMetricNamespace)
    .metricData(metricDataList)
    .build();

cw.putMetricData(request);
System.out.println("Added metric values for for metric " +
customMetricName);

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [PutMetricData](#)”中的。

场景

开始使用 CloudWatch 指标、控制面板和告警

以下代码示例展示了如何：

- 列出 CloudWatch 命名空间和指标。
- 获取指标和预估账单的统计数据。
- 创建和更新控制面板。
- 创建数据并将数据添加到指标。
- 创建并触发告警，然后查看告警历史记录。
- 添加异常检测器
- 获取指标映像，然后清理资源。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import com.fasterxml.jackson.core.JsonFactory;
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchClient;
import software.amazon.awssdk.services.cloudwatch.model.AlarmHistoryItem;
import software.amazon.awssdk.services.cloudwatch.model.AlarmType;
import software.amazon.awssdk.services.cloudwatch.model.AnomalyDetector;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatch.model.ComparisonOperator;
import software.amazon.awssdk.services.cloudwatch.model.DashboardValidationMessage;
import software.amazon.awssdk.services.cloudwatch.model.Datapoint;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsRequest;
import
    software.amazon.awssdk.services.cloudwatch.model.DeleteAnomalyDetectorRequest;
import software.amazon.awssdk.services.cloudwatch.model.DeleteDashboardsRequest;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmHistoryRequest;
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmHistoryResponse;
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsForMetricRequest;
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsForMetricResponse;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsRequest;
import software.amazon.awssdk.services.cloudwatch.model.DescribeAlarmsResponse;
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAnomalyDetectorsRequest;
import
    software.amazon.awssdk.services.cloudwatch.model.DescribeAnomalyDetectorsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricDataRequest;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricDataResponse;
```

```
import software.amazon.awssdk.services.cloudwatch.model.GetMetricStatisticsRequest;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricStatisticsResponse;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricWidgetImageRequest;
import
    software.amazon.awssdk.services.cloudwatch.model.GetMetricWidgetImageResponse;
import software.amazon.awssdk.services.cloudwatch.model.HistoryItemType;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsRequest;
import software.amazon.awssdk.services.cloudwatch.model.ListMetricsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Metric;
import software.amazon.awssdk.services.cloudwatch.model.MetricAlarm;
import software.amazon.awssdk.services.cloudwatch.model.MetricDataQuery;
import software.amazon.awssdk.services.cloudwatch.model.MetricDataResult;
import software.amazon.awssdk.services.cloudwatch.model.MetricDatum;
import software.amazon.awssdk.services.cloudwatch.model.MetricStat;
import software.amazon.awssdk.services.cloudwatch.model.PutAnomalyDetectorRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutDashboardRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutDashboardResponse;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricAlarmRequest;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricDataRequest;
import software.amazon.awssdk.services.cloudwatch.model.ScanBy;
import software.amazon.awssdk.services.cloudwatch.model.SingleMetricAnomalyDetector;
import software.amazon.awssdk.services.cloudwatch.model.StandardUnit;
import software.amazon.awssdk.services.cloudwatch.model.Statistic;
import software.amazon.awssdk.services.cloudwatch.paginators.ListDashboardsIterable;
import software.amazon.awssdk.services.cloudwatch.paginators.ListMetricsIterable;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.ZoneOffset;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```

*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
* To enable billing metrics and statistics for this example, make sure billing
* alerts are enabled for your account:
* https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/monitor\_estimated\_charges\_with\_cloudwatch.html#turning\_on\_billing\_metrics
*
* This Java code example performs the following tasks:
*
* 1. List available namespaces from Amazon CloudWatch.
* 2. List available metrics within the selected Namespace.
* 3. Get statistics for the selected metric over the last day.
* 4. Get CloudWatch estimated billing for the last week.
* 5. Create a new CloudWatch dashboard with metrics.
* 6. List dashboards using a paginator.
* 7. Create a new custom metric by adding data for it.
* 8. Add the custom metric to the dashboard.
* 9. Create an alarm for the custom metric.
* 10. Describe current alarms.
* 11. Get current data for the new custom metric.
* 12. Push data into the custom metric to trigger the alarm.
* 13. Check the alarm state using the action DescribeAlarmsForMetric.
* 14. Get alarm history for the new alarm.
* 15. Add an anomaly detector for the custom metric.
* 16. Describe current anomaly detectors.
* 17. Get a metric image for the custom metric.
* 18. Clean up the Amazon CloudWatch resources.
*/
public class CloudWatchScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException {
        final String usage = ""

            Usage:
                <myDate> <costDateWeek> <dashboardName> <dashboardJson>
<dashboardAdd> <settings> <metricImage> \s

            Where:
                myDate - The start date to use to get metric statistics. (For
example, 2023-01-11T18:35:24.00Z.)\s

```

```

        costDateWeek - The start date to use to get AWS/Billinget
statistics. (For example, 2023-01-11T18:35:24.00Z.)\s
        dashboardName - The name of the dashboard to create.\s
        dashboardJson - The location of a JSON file to use to create a
dashboard. (See Readme file.)\s
        dashboardAdd - The location of a JSON file to use to update a
dashboard. (See Readme file.)\s
        settings - The location of a JSON file from which various values
are read. (See Readme file.)\s
        metricImage - The location of a BMP file that is used to create a
graph.\s
        """;

    if (args.length != 7) {
        System.out.println(usage);
        System.exit(1);
    }

    Region region = Region.US_EAST_1;
    String myDate = args[0];
    String costDateWeek = args[1];
    String dashboardName = args[2];
    String dashboardJson = args[3];
    String dashboardAdd = args[4];
    String settings = args[5];
    String metricImage = args[6];

    Double dataPoint = Double.parseDouble("10.0");
    Scanner sc = new Scanner(System.in);
    CloudWatchClient cw = CloudWatchClient.builder()
        .region(region)
        .credentialsProvider(ProfileCredentialsProvider.create())
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the Amazon CloudWatch example scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println(
        "1. List at least five available unique namespaces from Amazon
CloudWatch. Select one from the list.");
    ArrayList<String> list = listNameSpaces(cw);
    for (int z = 0; z < 5; z++) {

```



```
        int index = z + 1;
        System.out.println("    " + index + ". " + list.get(z));
    }

    String selectedNamespace = "";
    String selectedMetrics = "";
    int num = Integer.parseInt(sc.nextLine());
    if (1 <= num && num <= 5) {
        selectedNamespace = list.get(num - 1);
    } else {
        System.out.println("You did not select a valid option.");
        System.exit(1);
    }
    System.out.println("You selected " + selectedNamespace);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("2. List available metrics within the selected namespace
and select one from the list.");
    ArrayList<String> metList = listMets(cw, selectedNamespace);
    for (int z = 0; z < 5; z++) {
        int index = z + 1;
        System.out.println("    " + index + ". " + metList.get(z));
    }
    num = Integer.parseInt(sc.nextLine());
    if (1 <= num && num <= 5) {
        selectedMetrics = metList.get(num - 1);
    } else {
        System.out.println("You did not select a valid option.");
        System.exit(1);
    }
    System.out.println("You selected " + selectedMetrics);
    Dimension myDimension = getSpecificMet(cw, selectedNamespace);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("3. Get statistics for the selected metric over the last
day.");
    String metricOption = "";
    ArrayList<String> statTypes = new ArrayList<>();
    statTypes.add("SampleCount");
    statTypes.add("Average");
    statTypes.add("Sum");
    statTypes.add("Minimum");
```

```
statTypes.add("Maximum");

for (int t = 0; t < 5; t++) {
    System.out.println("    " + (t + 1) + ". " + statTypes.get(t));
}
System.out.println("Select a metric statistic by entering a number from the
preceding list:");
num = Integer.parseInt(sc.nextLine());
if (1 <= num && num <= 5) {
    metricOption = statTypes.get(num - 1);
} else {
    System.out.println("You did not select a valid option.");
    System.exit(1);
}
System.out.println("You selected " + metricOption);
getAndDisplayMetricStatistics(cw, selectedNamespace, selectedMetrics,
metricOption, myDate, myDimension);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Get CloudWatch estimated billing for the last
week.");
getMetricStatistics(cw, costDateWeek);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Create a new CloudWatch dashboard with metrics.");
createDashboardWithMetrics(cw, dashboardName, dashboardJson);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. List dashboards using a paginator.");
listDashboards(cw);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Create a new custom metric by adding data to it.");
createNewCustomMetric(cw, dataPoint);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Add an additional metric to the dashboard.");
addMetricToDashboard(cw, dashboardAdd, dashboardName);
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("9. Create an alarm for the custom metric.");
String alarmName = createAlarm(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Describe ten current alarms.");
describeAlarms(cw);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Get current data for new custom metric.");
getCustomMetricData(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Push data into the custom metric to trigger the
alarm.");
addMetricDataForAlarm(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Check the alarm state using the action
DescribeAlarmsForMetric.");
checkForMetricAlarm(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Get alarm history for the new alarm.");
getAlarmHistory(cw, settings, myDate);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("15. Add an anomaly detector for the custom metric.");
addAnomalyDetector(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("16. Describe current anomaly detectors.");
describeAnomalyDetectors(cw, settings);
System.out.println(DASHES);

System.out.println(DASHES);
```

```
        System.out.println("17. Get a metric image for the custom metric.");
        getAndOpenMetricImage(cw, metricImage);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Clean up the Amazon CloudWatch resources.");
        deleteDashboard(cw, dashboardName);
        deleteCWAlarm(cw, alarmName);
        deleteAnomalyDetector(cw, settings);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Amazon CloudWatch example scenario is complete.");
        System.out.println(DASHES);
        cw.close();
    }

    public static void deleteAnomalyDetector(CloudWatchClient cw, String fileName) {
        try {
            // Read values from the JSON file.
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();

            SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                .metricName(customMetricName)
                .namespace(customMetricNamespace)
                .stat("Maximum")
                .build();

            DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
                .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                .build();

            cw.deleteAnomalyDetector(request);
            System.out.println("Successfully deleted the Anomaly Detector.");

        } catch (CloudWatchException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public static void deleteCWAlarm(CloudWatchClient cw, String alarmName) {
    try {
        DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
            .alarmNames(alarmName)
            .build();

        cw.deleteAlarms(request);
        System.out.println("Successfully deleted alarm " + alarmName);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteDashboard(CloudWatchClient cw, String dashboardName) {
    try {
        DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
            .dashboardNames(dashboardName)
            .build();
        cw.deleteDashboards(dashboardsRequest);
        System.out.println(dashboardName + " was successfully deleted.");

    } catch (CloudWatchException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void getAndOpenMetricImage(CloudWatchClient cw, String fileName) {
    System.out.println("Getting Image data for custom metric.");
    try {
        String myJSON = "{\n" +
            "  \"title\": \"Example Metric Graph\",\n" +
            "  \"view\": \"timeSeries\",\n" +
            "  \"stacked\": false,\n" +
```

```

        "  \"period\": 10,\n" +
        "  \"width\": 1400,\n" +
        "  \"height\": 600,\n" +
        "  \"metrics\": [\n" +
        "    [\n" +
        "      \"AWS/Billing\",\n" +
        "      \"EstimatedCharges\",\n" +
        "      \"Currency\",\n" +
        "      \"USD\"\n" +
        "    ]\n" +
        "  ]\n" +
        "];

    GetMetricWidgetImageRequest imageRequest =
    GetMetricWidgetImageRequest.builder()
        .metricWidget(myJSON)
        .build();

    GetMetricWidgetImageResponse response =
    cw.getMetricWidgetImage(imageRequest);
    SdkBytes sdkBytes = response.metricWidgetImage();
    byte[] bytes = sdkBytes.asByteArray();
    File outputFile = new File(fileName);
    try (FileOutputStream outputStream = new FileOutputStream(outputFile)) {
        outputStream.write(bytes);
    }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void describeAnomalyDetectors(CloudWatchClient cw, String
fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

```

```
DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
    .maxResults(10)
    .metricName(customMetricName)
    .namespace(customMetricNamespace)
    .build();

DescribeAnomalyDetectorsResponse response =
cw.describeAnomalyDetectors(detectorsRequest);
List<AnomalyDetector> anomalyDetectorList = response.anomalyDetectors();
for (AnomalyDetector detector : anomalyDetectorList) {
    System.out.println("Metric name: " +
detector.singleMetricAnomalyDetector().metricName());
    System.out.println("State: " + detector.stateValue());
}

} catch (CloudWatchException | IOException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

public static void addAnomalyDetector(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .stat("Maximum")
            .build();

        PutAnomalyDetectorRequest anomalyDetectorRequest =
PutAnomalyDetectorRequest.builder()
            .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
            .build();
```

```
        cw.putAnomalyDetector(anomalyDetectorRequest);
        System.out.println("Added anomaly detector for metric " +
customMetricName + ".");

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void getAlarmHistory(CloudWatchClient cw, String fileName, String
date) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String alarmName = rootNode.findValue("exampleAlarmName").asText();

        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();
        DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
            .startDate(start)
            .endDate(endDate)
            .alarmName(alarmName)
            .historyItemType(HistoryItemType.ACTION)
            .build();

        DescribeAlarmHistoryResponse response =
cw.describeAlarmHistory(historyRequest);
        List<AlarmHistoryItem> historyItems = response.alarmHistoryItems();
        if (historyItems.isEmpty()) {
            System.out.println("No alarm history data found for " + alarmName +
".");
        } else {
            for (AlarmHistoryItem item : historyItems) {
                System.out.println("History summary: " + item.historySummary());
                System.out.println("Time stamp: " + item.timestamp());
            }
        }
    }

    } catch (CloudWatchException | IOException e) {
```



```
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void checkForMetricAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        boolean hasAlarm = false;
        int retries = 10;

        DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        while (!hasAlarm && retries > 0) {
            DescribeAlarmsForMetricResponse response =
cw.describeAlarmsForMetric(metricRequest);
            hasAlarm = response.hasMetricAlarms();
            retries--;
            Thread.sleep(20000);
            System.out.println(".");
        }
        if (!hasAlarm)
            System.out.println("No Alarm state found for " + customMetricName +
" after 10 retries.");
        else
            System.out.println("Alarm state found for " + customMetricName +
".");

    } catch (CloudWatchException | IOException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
public static void addMetricDataForAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        // Set an Instant object.
        String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
        Instant instant = Instant.parse(time);

        MetricDatum datum = MetricDatum.builder()
            .metricName(customMetricName)
            .unit(StandardUnit.NONE)
            .value(1001.00)
            .timestamp(instant)
            .build();

        MetricDatum datum2 = MetricDatum.builder()
            .metricName(customMetricName)
            .unit(StandardUnit.NONE)
            .value(1002.00)
            .timestamp(instant)
            .build();

        List<MetricDatum> metricDataList = new ArrayList<>();
        metricDataList.add(datum);
        metricDataList.add(datum2);

        PutMetricDataRequest request = PutMetricDataRequest.builder()
            .namespace(customMetricNamespace)
            .metricData(metricDataList)
            .build();

        cw.putMetricData(request);
        System.out.println("Added metric values for for metric " +
customMetricName);
    }
}
```

```
    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void getCustomMetricData(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();

        // Set the date.
        Instant nowDate = Instant.now();

        long hours = 1;
        long minutes = 30;
        Instant date2 = nowDate.plus(hours, ChronoUnit.HOURS).plus(minutes,
ChronoUnit.MINUTES);

        Metric met = Metric.builder()
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .build();

        MetricStat metStat = MetricStat.builder()
            .stat("Maximum")
            .period(1)
            .metric(met)
            .build();

        MetricDataQuery dataQuery = MetricDataQuery.builder()
            .metricStat(metStat)
            .id("foo2")
            .returnData(true)
            .build();

        List<MetricDataQuery> dq = new ArrayList<>();
        dq.add(dataQuery);
    }
}
```

```
    GetMetricDataRequest getMetReq = GetMetricDataRequest.builder()
        .maxDatapoints(10)
        .scanBy(ScanBy.TIMESTAMP_DESCENDING)
        .startTime(nowDate)
        .endTime(date2)
        .metricDataQueries(dq)
        .build();

    GetMetricDataResponse response = cw.getMetricData(getMetReq);
    List<MetricDataResult> data = response.metricDataResults();
    for (MetricDataResult item : data) {
        System.out.println("The label is " + item.label());
        System.out.println("The status code is " +
item.statusCode().toString());
    }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void describeAlarms(CloudWatchClient cw) {
    try {
        List<AlarmType> typeList = new ArrayList<>();
        typeList.add(AlarmType.METRIC_ALARM);

        DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
            .alarmTypes(typeList)
            .maxRecords(10)
            .build();

        DescribeAlarmsResponse response = cw.describeAlarms(alarmsRequest);
        List<MetricAlarm> alarmList = response.metricAlarms();
        for (MetricAlarm alarm : alarmList) {
            System.out.println("Alarm name: " + alarm.alarmName());
            System.out.println("Alarm description: " +
alarm.alarmDescription());
        }
    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}

public static String createAlarm(CloudWatchClient cw, String fileName) {
    try {
        // Read values from the JSON file.
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName =
rootNode.findValue("customMetricName").asText();
        String alarmName = rootNode.findValue("exampleAlarmName").asText();
        String emailTopic = rootNode.findValue("emailTopic").asText();
        String accountId = rootNode.findValue("accountId").asText();
        String region = rootNode.findValue("region").asText();

        // Create a List for alarm actions.
        List<String> alarmActions = new ArrayList<>();
        alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);
        PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
            .alarmActions(alarmActions)
            .alarmDescription("Example metric alarm")
            .alarmName(alarmName)

.comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
            .threshold(100.00)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .evaluationPeriods(1)
            .period(10)
            .statistic("Maximum")
            .datapointsToAlarm(1)
            .treatMissingData("ignore")
            .build();

        cw.putMetricAlarm(alarmRequest);
        System.out.println(alarmName + " was successfully created!");
        return alarmName;

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
    }
    return "";
}

public static void addMetricToDashboard(CloudWatchClient cw, String fileName,
String dashboardName) {
    try {
        PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
            .dashboardName(dashboardName)
            .dashboardBody(readFileAsString(fileName))
            .build();

        cw.putDashboard(dashboardRequest);
        System.out.println(dashboardName + " was successfully updated.");

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void createNewCustomMetric(CloudWatchClient cw, Double dataPoint)
{
    try {
        Dimension dimension = Dimension.builder()
            .name("UNIQUE_PAGES")
            .value("URLS")
            .build();

        // Set an Instant object.
        String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
        Instant instant = Instant.parse(time);

        MetricDatum datum = MetricDatum.builder()
            .metricName("PAGES_VISITED")
            .unit(StandardUnit.NONE)
            .value(dataPoint)
            .timestamp(instant)
            .dimensions(dimension)
            .build();

        PutMetricDataRequest request = PutMetricDataRequest.builder()
            .namespace("SITE/TRAFFIC")
```

```
        .metricData(datum)
        .build();

        cw.putMetricData(request);
        System.out.println("Added metric values for for metric PAGES_VISITED");

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void listDashboards(CloudWatchClient cw) {
    try {
        ListDashboardsIterable listRes = cw.listDashboardsPaginator();
        listRes.stream()
            .flatMap(r -> r.dashboardEntries().stream())
            .forEach(entry -> {
                System.out.println("Dashboard name is: " +
entry.dashboardName());
                System.out.println("Dashboard ARN is: " +
entry.dashboardArn());
            });

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createDashboardWithMetrics(CloudWatchClient cw, String
dashboardName, String fileName) {
    try {
        PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
            .dashboardName(dashboardName)
            .dashboardBody(readFileAsString(fileName))
            .build();

        PutDashboardResponse response = cw.putDashboard(dashboardRequest);
        System.out.println(dashboardName + " was successfully created.");
        List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
        if (messages.isEmpty()) {
            System.out.println("There are no messages in the new Dashboard");
        }
    }
}
```

```
        } else {
            for (DashboardValidationMessage message : messages) {
                System.out.println("Message is: " + message.message());
            }
        }

    } catch (CloudWatchException | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static String readFileAsString(String file) throws IOException {
    return new String(Files.readAllBytes(Paths.get(file)));
}

public static void getMetricStatistics(CloudWatchClient cw, String costDateWeek)
{
    try {
        Instant start = Instant.parse(costDateWeek);
        Instant endDate = Instant.now();
        Dimension dimension = Dimension.builder()
            .name("Currency")
            .value("USD")
            .build();

        List<Dimension> dimensionList = new ArrayList<>();
        dimensionList.add(dimension);
        GetMetricStatisticsRequest statisticsRequest =
        GetMetricStatisticsRequest.builder()
            .metricName("EstimatedCharges")
            .namespace("AWS/Billing")
            .dimensions(dimensionList)
            .statistics(Statistic.MAXIMUM)
            .startTime(start)
            .endTime(endDate)
            .period(86400)
            .build();

        GetMetricStatisticsResponse response =
        cw.getMetricStatistics(statisticsRequest);
        List<Datapoint> data = response.datapoints();
        if (!data.isEmpty()) {
            for (Datapoint datapoint : data) {
```



```
        System.out
            .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
    }
    } else {
        System.out.println("The returned data list is empty");
    }

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getAndDisplayMetricStatistics(CloudWatchClient cw, String
nameSpace, String metVal,
String metricOption, String date, Dimension myDimension) {
    try {
        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();

        GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
        .endTime(endDate)
        .startTime(start)
        .dimensions(myDimension)
        .metricName(metVal)
        .namespace(nameSpace)
        .period(86400)
        .statistics(Statistic.fromValue(metricOption))
        .build();

        GetMetricStatisticsResponse response =
cw.getMetricStatistics(statisticsRequest);
        List<Datapoint> data = response.datapoints();
        if (!data.isEmpty()) {
            for (Datapoint datapoint : data) {
                System.out
                    .println("Timestamp: " + datapoint.timestamp() + "
Maximum value: " + datapoint.maximum());
            }
        } else {
            System.out.println("The returned data list is empty");
        }
    }
}
```

```
    } catch (CloudWatchException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static Dimension getSpecificMet(CloudWatchClient cw, String namespace) {
    try {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        ListMetricsResponse response = cw.listMetrics(request);
        List<Metric> myList = response.metrics();
        Metric metric = myList.get(0);
        return metric.dimensions().get(0);

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

public static ArrayList<String> listMets(CloudWatchClient cw, String namespace)
{
    try {
        ArrayList<String> metList = new ArrayList<>();
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        ListMetricsIterable listRes = cw.listMetricsPaginator(request);
        listRes.stream()
            .flatMap(r -> r.metrics().stream())
            .forEach(metrics -> metList.add(metrics.metricName()));

        return metList;

    } catch (CloudWatchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
        return null;
    }

    public static ArrayList<String> listNameSpaces(CloudWatchClient cw) {
        try {
            ArrayList<String> nameSpaceList = new ArrayList<>();
            ListMetricsRequest request = ListMetricsRequest.builder()
                .build();

            ListMetricsIterable listRes = cw.listMetricsPaginator(request);
            listRes.stream()
                .flatMap(r -> r.metrics().stream())
                .forEach(metrics -> {
                    String data = metrics.namespace();
                    if (!nameSpaceList.contains(data)) {
                        nameSpaceList.add(data);
                    }
                });

            return nameSpaceList;
        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [DeleteAlarms](#)
 - [DeleteAnomalyDetector](#)
 - [DeleteDashboards](#)
 - [DescribeAlarmHistory](#)
 - [DescribeAlarms](#)
 - [DescribeAlarmsForMetric](#)
 - [DescribeAnomalyDetectors](#)
 - [GetMetricData](#)
 - [GetMetricStatistics](#)

- [GetMetricWidgetImage](#)
- [ListMetrics](#)
- [PutAnomalyDetector](#)
- [PutDashboard](#)
- [PutMetricAlarm](#)
- [PutMetricData](#)

CloudWatch SDK适用于 Java 2.x 的事件示例

以下代码示例向您展示了如何使用 with Events 来执行操作和实现常见场景。AWS SDK for Java 2.x CloudWatch

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

PutEvents

以下代码示例演示如何使用 PutEvents。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [PutEvents](#)”中的。

PutRule

以下代码示例演示如何使用 PutRule。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [PutRule](#)”中的。

PutTargets

以下代码示例演示如何使用 PutTargets。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);  
        }  
    }  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [PutTargets](#)”中的。

CloudWatch 使用SDK适用于 Java 2.x 的日志示例

以下代码示例向您展示了如何使用 with Logs 来执行操作和实现常见场 CloudWatch 景。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

DeleteSubscriptionFilter

以下代码示例演示如何使用 DeleteSubscriptionFilter。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteSubscriptionFilter](#)”中的。

DescribeSubscriptionFilters

以下代码示例演示如何使用 DescribeSubscriptionFilters。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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");
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeSubscriptionFilters](#)”中的。

PutSubscriptionFilter

以下代码示例演示如何使用 PutSubscriptionFilter。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [PutSubscriptionFilter](#)”中的。

StartLiveTail

以下代码示例演示如何使用 StartLiveTail。

SDK适用于 Java 2.x

包含所需的文件。

```
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;
```

处理 Live Tail 会话中的事件。

```
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();
}
```

启动 Live Tail 会话。

```
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));
```

经过一段时间后停止 Live Tail 会话。

```
/* 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");
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartLiveTail](#)”中的。

SDK适用于 Java 2.x 的 Amazon Cognito 身份示例

以下代码示例向您展示了如何使用 AWS SDK for Java 2.x 与 Amazon Cognito Identity 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

CreateIdentityPool

以下代码示例演示如何使用 CreateIdentityPool。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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.CognitoIdentityProviderException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * 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 "";
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateIdentityPool](#)”中的。

DeleteIdentityPool

以下代码示例演示如何使用 DeleteIdentityPool。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 identityPoolId = args[0];
        CognitoIdentityClient cognitoIdClient = CognitoIdentityClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();

        deleteIdPool(cognitoIdClient, identityPoolId);
        cognitoIdClient.close();
    }
}
```

```
public static void deleteIdPool(CognitoIdentityClient cognitoIdClient, String
identityPoolId) {
    try {

        DeleteIdentityPoolRequest identityPoolRequest =
DeleteIdentityPoolRequest.builder()
        .identityPoolId(identityPoolId)
        .build();

        cognitoIdClient.deleteIdentityPool(identityPoolRequest);
        System.out.println("Done");

    } catch (AwsServiceException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteIdentityPool](#)”中的。

GetCredentialsForIdentity

以下代码示例演示如何使用 `GetCredentialsForIdentity`。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 getCredsForIdentityRequest =
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetCredentialsForIdentity](#)”中的。

ListIdentityPools

以下代码示例演示如何使用 ListIdentityPools。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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.CognitoIdentityProviderException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```

```
* 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);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListIdentityPools](#)”中的。

SDK适用于 Java 2.x 的 Amazon Cognito 身份提供商示例

以下代码示例向您展示了如何使用 AWS SDK for Java 2.x 与 Amazon Cognito 身份提供商配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

开始使用

开始使用 Amazon Cognito

以下代码示例展示了如何开始使用 Amazon Cognito。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListUserPools](#)”中的。

主题

- [操作](#)
- [场景](#)

操作

AdminGetUser

以下代码示例演示如何使用 AdminGetUser。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [AdminGetUser](#)”中的。

AdminInitiateAuth

以下代码示例演示如何使用 AdminInitiateAuth。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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;
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [AdminInitiateAuth](#)”中的。

AdminRespondToAuthChallenge

以下代码示例演示如何使用 AdminRespondToAuthChallenge。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

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

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [AdminRespondToAuthChallenge](#)”中的。

AssociateSoftwareToken

以下代码示例演示如何使用 AssociateSoftwareToken。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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();
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [AssociateSoftwareToken](#)”中的。

ConfirmSignUp

以下代码示例演示如何使用 ConfirmSignUp。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ConfirmSignUp](#)”中的。

CreateUserPool

以下代码示例演示如何使用 CreateUserPool。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
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 "";
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateUserPool!](#)”中的。

CreateUserPoolClient

以下代码示例演示如何使用 CreateUserPoolClient。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateUserPoolClient](#)”中的。

ListUserPools

以下代码示例演示如何使用 ListUserPools。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListUserPools](#)”中的。

ListUsers

以下代码示例演示如何使用 ListUsers。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import
    software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderException;
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListUsers](#)”中的。

ResendConfirmationCode

以下代码示例演示如何使用 ResendConfirmationCode。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ResendConfirmationCode](#)”中的。

SignUp

以下代码示例演示如何使用 SignUp。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [SignUp](#)”中的。

VerifySoftwareToken

以下代码示例演示如何使用 VerifySoftwareToken。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

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

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [VerifySoftwareToken](#)”中的。

场景

使用需要的用户池注册用户 MFA

以下代码示例展示了如何：

- 使用用户名、密码和电子邮件地址注册和确认用户。
- 通过将MFA应用程序与用户关联来设置多因素身份验证。
- 使用密码和密码登录。MFA

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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.AssociateSoftwareTokenResponse;
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.CognitoIdentityProviderException;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ConfirmSignUpRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeRequest;
import
    software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeResponse;
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("**** Confirmation 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);
    }
}
}
```

• 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。

- [AdminGetUser](#)
- [AdminInitiateAuth](#)
- [AdminRespondToAuthChallenge](#)
- [AssociateSoftwareToken](#)
- [ConfirmDevice](#)
- [ConfirmSignUp](#)
- [InitiateAuth](#)
- [ListUsers](#)
- [ResendConfirmationCode](#)
- [RespondToAuthChallenge](#)
- [SignUp](#)
- [VerifySoftwareToken](#)

用于 Java 2.x 的 Amazon Comprehend SDK d 示例

以下代码示例向您展示了如何使用 AWS SDK for Java 2.x 与 Amazon Comprehend 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

CreateDocumentClassifier

以下代码示例演示如何使用 CreateDocumentClassifier。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateDocumentClassifier](#)”中的。

DetectDominantLanguage

以下代码示例演示如何使用 DetectDominantLanguage。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

```
    }  
  }  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DetectDominantLanguage](#)”中的。

DetectEntities

以下代码示例演示如何使用 DetectEntities。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DetectEntities](#)”中的。

DetectKeyPhrases

以下代码示例演示如何使用 DetectKeyPhrases。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DetectKeyPhrases](#)”中的。

DetectSentiment

以下代码示例演示如何使用 DetectSentiment。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DetectSentiment](#)”中的。

DetectSyntax

以下代码示例演示如何使用 DetectSyntax。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DetectSyntax](#)”中的。

使用适用于 Java 2.x 的 DynamoDB 示例 SDK

以下代码示例向您展示了如何在 DynamoDB 中使用来执行操作和实现常见场景。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

开始使用

开始使用 DynamoDB

以下代码示例演示如何开始使用 DynamoDB。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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!");
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListTables](#)”中的。

主题

- [操作](#)
- [场景](#)
- [无服务器示例](#)

操作

BatchGetItem

以下代码示例演示如何使用 BatchGetItem。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

展示如何使用服务客户端获取批量项目。

```
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.");
        }
    }
}

```

展示如何使用服务客户端和分页工具获取批量项目。

```

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());
        });
}
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [BatchGetItem](#)”中的。

BatchWriteItem

以下代码示例演示如何使用 BatchWriteItem。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用服务客户端将许多项目插入到表中。

```
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(item1Attributes)

        // 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(item2Attributes)

        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);
    }
}
}
```

使用增强型客户端将许多项目插入到表中。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [BatchWriteItem](#)”中的。

CreateTable

以下代码示例演示如何使用 CreateTable。

SDK适用于 Java 2.x

Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```

```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class CreateTable {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <tableName> <key>

            Where:
                tableName - The Amazon DynamoDB table to create (for example,
Music3).
                key - The key for the Amazon DynamoDB table (for example,
Artist).

            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        System.out.println("Creating an Amazon DynamoDB table " + tableName + " with
a simple primary key: " + key);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        String result = createTable(ddb, tableName, key);
        System.out.println("New table is " + result);
        ddb.close();
    }

    public static String createTable(DynamoDbClient ddb, String tableName, String
key) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        CreateTableRequest request = CreateTableRequest.builder()
            .attributeDefinitions(AttributeDefinition.builder()
                .attributeName(key)
```

```
        .attributeType(ScalarAttributeType.S)
        .build()
    .keySchema(KeySchemaElement.builder()
        .attributeName(key)
        .keyType(KeyType.HASH)
        .build())
    .provisionedThroughput(ProvisionedThroughput.builder()
        .readCapacityUnits(10L)
        .writeCapacityUnits(10L)
        .build())
    .tableName(tableName)
    .build();

String newTable;
try {
    CreateTableResponse response = ddb.createTable(request);
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    // Wait until the Amazon DynamoDB table is created.
    WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    newTable = response.tableDescription().tableName();
    return newTable;

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateTable](#)”中的。

DeleteItem

以下代码示例演示如何使用 DeleteItem。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteItem](#)”中的。

DeleteTable

以下代码示例演示如何使用 DeleteTable。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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!");
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteTable](#)”中的。

DescribeTable

以下代码示例演示如何使用 DescribeTable。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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();

        describeDynamoDBTable(ddb, tableName);
        ddb.close();
    }
}
```

```
public static void describeDynamoDBTable(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!");
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeTable](#)”中的。

DescribeTimeToLive

以下代码示例演示如何使用 DescribeTimeToLive。

SDK适用于 Java 2.x

描述现有 DynamoDB 表的TTL配置。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DescribeTimeToLiveRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTimeToLiveResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.Optional;

    final DescribeTimeToLiveRequest request =
DescribeTimeToLiveRequest.builder()
    .tableName(tableName)
    .build();
    try (DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build()) {
        final DescribeTimeToLiveResponse response =
ddb.describeTimeToLive(request);
        System.out.println(tableName + " description of time to live is "
            + response.toString());
    } catch (ResourceNotFoundException e) {
        System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.exit(0);
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeTimeToLive](#)”中的。

GetItem

以下代码示例演示如何使用 GetItem。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用从表中获取项目 DynamoDbClient。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To get an item from an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client, see the EnhancedGetItem example.
 */
public class GetItem {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <tableName> <key> <keyVal>

                Where:
```

```

        tableName - The Amazon DynamoDB table from which an item is
retrieved (for example, Music3).\s
        key - The key used in the Amazon DynamoDB table (for example,
Artist).\s
        keyval - The key value that represents the item to get (for
example, Famous Band).
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String tableName = args[0];
    String key = args[1];
    String keyVal = args[2];
    System.out.format("Retrieving item \"%s\" from \"%s\"\n", keyVal,
tableName);
    Region region = Region.US_EAST_1;
    DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build();

    getDynamoDBItem(ddb, tableName, key, keyVal);
    ddb.close();
}

public static void getDynamoDBItem(DynamoDbClient ddb, String tableName, String
key, String keyVal) {
    HashMap<String, AttributeValue> keyToGet = new HashMap<>();
    keyToGet.put(key, AttributeValue.builder()
        .s(keyVal)
        .build());

    GetItemRequest request = GetItemRequest.builder()
        .key(keyToGet)
        .tableName(tableName)
        .build();

    try {
        // If there is no matching item, GetItem does not return any data.
        Map<String, AttributeValue> returnedItem = ddb.getItem(request).item();
        if (returnedItem.isEmpty())
            System.out.format("No item found with the key %s!\n", key);
    }
}

```



```
        else {
            Set<String> keys = returnedItem.keySet();
            System.out.println("Amazon DynamoDB table attributes: \n");
            for (String key1 : keys) {
                System.out.format("%s: %s\n", key1,
returnedItem.get(key1).toString());
            }
        }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetItem](#)”中的。

ListTables

以下代码示例演示如何使用 ListTables。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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!");  
}  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListTables](#)”中的。

PutItem

以下代码示例演示如何使用 PutItem。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用将项目放入表格中 [DynamoDbClient](#)。

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;  
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;  
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;  
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;  
import software.amazon.awssdk.services.dynamodb.model.PutItemResponse;  
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;  
import java.util.HashMap;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic: */
```

```

*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
* To place items into an Amazon DynamoDB table using the AWS SDK for Java V2,
* its better practice to use the
* Enhanced Client. See the EnhancedPutItem example.
*/
public class PutItem {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                    <tableName> <key> <keyVal> <albumtitle> <albumtitleval> <awards>
<awardsval> <Songtitle> <songtitleval>

                Where:
                    tableName - The Amazon DynamoDB table in which an item is placed
(for example, Music3).
                    key - The key used in the Amazon DynamoDB table (for example,
Artist).
                    keyval - The key value that represents the item to get (for
example, Famous Band).
                    albumTitle - The Album title (for example, AlbumTitle).
                    AlbumTitleValue - The name of the album (for example, Songs
About Life ).
                    Awards - The awards column (for example, Awards).
                    AwardVal - The value of the awards (for example, 10).
                    SongTitle - The song title (for example, SongTitle).
                    SongTitleVal - The value of the song title (for example, Happy
Day).

                **Warning** This program will place an item that you specify into a
table!

                """;

        if (args.length != 9) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String key = args[1];
        String keyVal = args[2];
        String albumTitle = args[3];
        String albumTitleValue = args[4];

```

```
String awards = args[5];
String awardVal = args[6];
String songTitle = args[7];
String songTitleVal = args[8];

Region region = Region.US_EAST_1;
DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build();

putItemInTable(ddb, tableName, key, keyVal, albumTitle, albumTitleValue,
awards, awardVal, songTitle,
    songTitleVal);
System.out.println("Done!");
ddb.close();
}

public static void putItemInTable(DynamoDbClient ddb,
    String tableName,
    String key,
    String keyVal,
    String albumTitle,
    String albumTitleValue,
    String awards,
    String awardVal,
    String songTitle,
    String songTitleVal) {

    HashMap<String, AttributeValue> itemValues = new HashMap<>();
    itemValues.put(key, AttributeValue.builder().s(keyVal).build());
    itemValues.put(songTitle, AttributeValue.builder().s(songTitleVal).build());
    itemValues.put(albumTitle,
AttributeValue.builder().s(albumTitleValue).build());
    itemValues.put(awards, AttributeValue.builder().s(awardVal).build());

    PutItemRequest request = PutItemRequest.builder()
        .tableName(tableName)
        .item(itemValues)
        .build();

    try {
        PutItemResponse response = ddb.putItem(request);
        System.out.println(tableName + " was successfully updated. The request
id is "
```

```

        + response.responseMetadata().requestId());

    } catch (ResourceNotFoundException e) {
        System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
        System.err.println("Be sure that it exists and that you've typed its
name correctly!");
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [PutItem](#)”中的。

Query

以下代码示例演示如何使用 Query。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用查询表 [DynamoDbClient](#)。

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.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;
}
}
```

使用 `DynamoDbClient` 和二级索引查询表。


```
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(expressionAttributeNames)
        .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);
}
}
```

- 有关API详细信息，请参阅AWS SDK for Java 2.x API参考文献中的[查询](#)。

Scan

以下代码示例演示如何使用 Scan。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用扫描 Amazon DynamoDB 表。 [DynamoDbClient](#)

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

```
import software.amazon.awssdk.services.dynamodb.model.ScanRequest;
import software.amazon.awssdk.services.dynamodb.model.ScanResponse;
import java.util.Map;
import java.util.Set;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To scan items from an Amazon DynamoDB table using the AWS SDK for Java V2,
 * its better practice to use the
 * Enhanced Client, See the EnhancedScanRecords example.
 */

public class DynamoDBScanItems {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <tableName>

            Where:
                tableName - The Amazon DynamoDB table to get information from
(for example, Music3).
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        scanItems(ddb, tableName);
        ddb.close();
    }
}
```

```
}

public static void scanItems(DynamoDbClient ddb, String tableName) {
    try {
        ScanRequest scanRequest = ScanRequest.builder()
            .tableName(tableName)
            .build();

        ScanResponse response = ddb.scan(scanRequest);
        for (Map<String, AttributeValue> item : response.items()) {
            Set<String> keys = item.keySet();
            for (String key : keys) {
                System.out.println("The key name is " + key + "\n");
                System.out.println("The value is " + item.get(key).s());
            }
        }

    } catch (DynamoDbException e) {
        e.printStackTrace();
        System.exit(1);
    }
}
}
```

- 有关API详细信息，请参阅 [Scan in AWS SDK for Java 2.x API Reference](#)。

UpdateItem

以下代码示例演示如何使用 UpdateItem。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用更新表中的项目 [DynamoDbClient](#)。

```
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!");
}
```

```
}  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [UpdateItem](#)”中的。

UpdateTimeToLive

以下代码示例演示如何使用 UpdateTimeToLive。

SDK适用于 Java 2.x

在现有 DynamoDB 表TTL上启用。

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;  
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;  
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;  
import software.amazon.awssdk.services.dynamodb.model.TimeToLiveSpecification;  
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveRequest;  
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveResponse;  
  
import java.util.Optional;  
  
    final TimeToLiveSpecification ttlSpecification =  
    TimeToLiveSpecification.builder()  
        .attributeName(ttlAttributeName)  
        .enabled(true)  
        .build();  
    final UpdateTimeToLiveRequest request = UpdateTimeToLiveRequest.builder()  
        .tableName(tableName)  
        .timeToLiveSpecification(ttlSpecification)  
        .build();  
    try (DynamoDbClient ddb = DynamoDbClient.builder()  
        .region(region)  
        .build()) {  
        final UpdateTimeToLiveResponse response =  
        ddb.updateTimeToLive(request);  
        System.out.println(tableName + " had its TTL successfully updated.  
The request id is "  
            + response.responseMetadata().requestId());  
    } catch (ResourceNotFoundException e) {
```

```

        System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Done!");

```

在现有 DynamoDB 表 TTL 上禁用。

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.TimeToLiveSpecification;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveResponse;

import java.util.Optional;

    final Region region = Optional.ofNullable(args[2]).isEmpty() ?
Region.US_EAST_1 : Region.of(args[2]);
    final TimeToLiveSpecification ttlSpecification =
TimeToLiveSpecification.builder()
        .attributeName(ttlAttributeName)
        .enabled(false)
        .build();
    final UpdateTimeToLiveRequest request = UpdateTimeToLiveRequest.builder()
        .tableName(tableName)
        .timeToLiveSpecification(ttlSpecification)
        .build();
    try (DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build()) {
        final UpdateTimeToLiveResponse response = ddb.updateTimeToLive(request);
        System.out.println(tableName + " had its TTL successfully updated. The
request id is "
            + response.responseMetadata().requestId());
    } catch (ResourceNotFoundException e) {
        System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);

```



```
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Done!");
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [UpdateTimeToLive](#)”中的。

场景

有条件地更新物品的 TTL

以下代码示例显示了如何有条件地更新项目。TTL

SDK适用于 Java 2.x

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package com.amazon.samplelib.ttl;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;
import software.amazon.awssdk.utils.ImmutableMap;

import java.util.Map;
import java.util.Optional;

public class UpdateTTLConditional {
    public static void main(String[] args) {
        final String usage = ""
            Usage:
                <tableName> <primaryKey> <sortKey> <newTtlAttribute> <region>
            Where:
                tableName - The Amazon DynamoDB table being queried.
                primaryKey - The name of the primary key. Also known as the hash
or partition key.
```

```

        sortKey - The name of the sort key. Also known as the range
attribute.
        newTtlAttribute - New attribute name (as part of the update
command)
        region (optional) - The AWS region that the Amazon DynamoDB
table is located in. (Default: us-east-1)
        """;
    // Optional "region" parameter - if args list length is NOT 3 or 4, short-
circuit exit.
    if (!(args.length == 4 || args.length == 5)) {
        System.out.println(usage);
        System.exit(1);
    }
    final String tableName = args[0];
    final String primaryKey = args[1];
    final String sortKey = args[2];
    final String newTtlAttribute = args[3];
    Region region = Optional.ofNullable(args[4]).isEmpty() ? Region.US_EAST_1 :
Region.of(args[4]);

    // Get current time in epoch second format
    final long currentTime = System.currentTimeMillis() / 1000;
    // Calculate expiration time 90 days from now in epoch second format
    final long expireDate = currentTime + (90 * 24 * 60 * 60);
    // An expression that defines one or more attributes to be updated, the
action to be performed on them, and new values for them.
    final String updateExpression = "SET newTtlAttribute = :val1";
    // A condition that must be satisfied in order for a conditional update to
succeed.
    final String conditionExpression = "expireAt > :val2";

    final ImmutableMap<String, AttributeValue> keyMap =
        ImmutableMap.of("primaryKey", AttributeValue.fromS(primaryKey),
            "sortKey", AttributeValue.fromS(sortKey));
    final Map<String, AttributeValue> expressionAttributeValues =
ImmutableMap.of(
        ":val1", AttributeValue.builder().s(newTtlAttribute).build(),
        ":val2",
        AttributeValue.builder().s(String.valueOf(expireDate)).build()
    );

    final UpdateItemRequest request = UpdateItemRequest.builder()
        .tableName(tableName)
        .key(keyMap)

```

```

        .updateExpression(updateExpression)
        .conditionExpression(conditionExpression)
        .expressionAttributeValues(expressionAttributeValues)
        .build();
    try (DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build()) {
        final UpdateItemResponse response = ddb.updateItem(request);
        System.out.println(tableName + " UpdateItem operation with conditional
TTL successful. Request id is "
            + response.responseMetadata().requestId());
    } catch (ResourceNotFoundException e) {
        System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.exit(0);
}
}
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [UpdateItem](#)”中的。

使用创建项目 TTL

以下代码示例显示了如何使用创建项目TTL。

SDK适用于 Java 2.x

```

// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

package com.amazon.samplelib.ttl;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.PutItemResponse;

```

```
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.utils.ImmutableMap;

import java.io.Serializable;
import java.util.Map;
import java.util.Optional;

public class CreateTTL {
    public static void main(String[] args) {
        final String usage = ""
            Usage:
                <tableName> <primaryKey> <sortKey> <region>
            Where:
                tableName - The Amazon DynamoDB table being queried.
                primaryKey - The name of the primary key. Also known as the hash
or partition key.
                sortKey - The name of the sort key. Also known as the range
attribute.
                region (optional) - The AWS region that the Amazon DynamoDB
table is located in. (Default: us-east-1)
            """;
        // Optional "region" parameter - if args list length is NOT 3 or 4, short-
circuit exit.
        if (!(args.length == 3 || args.length == 4)) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        String primaryKey = args[1];
        String sortKey = args[2];
        Region region = Optional.ofNullable(args[3]).isEmpty() ? Region.US_EAST_1 :
Region.of(args[3]);

        // Get current time in epoch second format
        final long createDate = System.currentTimeMillis() / 1000;

        // Calculate expiration time 90 days from now in epoch second format
        final long expireDate = createDate + (90 * 24 * 60 * 60);

        final ImmutableMap<String, ? extends Serializable> itemMap =
            ImmutableMap.of("primaryKey", primaryKey,
                "sortKey", sortKey,
                "creationDate", createDate,
```

```
        "expireAt", expireDate);
    final PutItemRequest request = PutItemRequest.builder()
        .tableName(tableName)
        .item((Map<String, AttributeValue>) itemMap)
        .build();
    try (DynamoDbClient ddb = DynamoDbClient.builder()
        .region(region)
        .build()) {
        final PutItemResponse response = ddb.putItem(request);
        System.out.println(tableName + " PutItem operation with TTL successful.
Request id is "
            + response.responseMetadata().requestId());
    } catch (ResourceNotFoundException e) {
        System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.exit(0);
}
```


- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [PutItem](#)”中的。

开始使用表、项目和查询

以下代码示例展示了如何：

- 创建可保存电影数据的表。
- 在表中加入单一电影，获取并更新此电影。
- 将样本JSON文件中的影片数据写入表中。
- 查询在给定年份发行的电影。
- 扫描在年份范围内发行的电影。
- 删除表中的电影后再删除表。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

创建 DynamoDB 表。

```
// Create a table with a Sort key.
public static void createTable(DynamoDbClient ddb, String tableName) {
    DynamoDbWaiter dbWaiter = ddb.waiter();
    ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

    // Define attributes.
    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("year")
        .attributeType("N")
        .build());

    attributeDefinitions.add(AttributeDefinition.builder()
        .attributeName("title")
        .attributeType("S")
        .build());

    ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
    KeySchemaElement key = KeySchemaElement.builder()
        .attributeName("year")
        .keyType(KeyType.HASH)
        .build();

    KeySchemaElement key2 = KeySchemaElement.builder()
        .attributeName("title")
        .keyType(KeyType.RANGE)
        .build();

    // Add KeySchemaElement objects to the list.
    tableKey.add(key);
    tableKey.add(key2);

    CreateTableRequest request = CreateTableRequest.builder()
        .keySchema(tableKey)
```

```

        .provisionedThroughput(ProvisionedThroughput.builder()
            .readCapacityUnits(10L)
            .writeCapacityUnits(10L)
            .build())
        .attributeDefinitions(attributeDefinitions)
        .tableName(tableName)
        .build();

    try {
        CreateTableResponse response = ddb.createTable(request);
        DescribeTableRequest tableRequest = DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        // Wait until the Amazon DynamoDB table is created.
        WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        String newTable = response.tableDescription().tableName();
        System.out.println("The " + newTable + " was successfully created.");

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

```

创建辅助函数来下载和解压缩示例JSON文件。

```

// 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++;
}
}
```

从表中获取项目。

```
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);
}
}
```

完整示例。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java example performs these tasks:
 *
 * 1. Creates the Amazon DynamoDB Movie table with partition and sort key.
 * 2. Puts data into the Amazon DynamoDB table from a JSON document using the
 * Enhanced client.
 * 3. Gets data from the Movie table.
 * 4. Adds a new item.
 * 5. Updates an item.
 * 6. Uses a Scan to query items using the Enhanced client.
 * 7. Queries all items where the year is 2013 using the Enhanced Client.
 * 8. Deletes the table.
 */
```

```
public class Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException {
        final String usage = ""

            Usage:
                <fileName>

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

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = "Movies";
        String fileName = args[0];
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon DynamoDB example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
            "1. Creating an Amazon DynamoDB table named Movies with a key named
year and a sort key named title.");
        createTable(ddb, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Loading data into the Amazon DynamoDB table.");
        loadData(ddb, tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
    }
}
```

```
        System.out.println("3. Getting data from the Movie table.");
        getItem(ddb);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Putting a record into the Amazon DynamoDB table.");
        putRecord(ddb);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Updating a record.");
        updateTableItem(ddb, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Scanning the Amazon DynamoDB table.");
        scanMovies(ddb, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Querying the Movies released in 2013.");
        queryTable(ddb);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        System.out.println(DASHES);

        ddb.close();
    }

    // Create a table with a Sort key.
    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
```

```
        .attributeName("title")
        .attributeType("S")
        .build());

ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
KeySchemaElement key = KeySchemaElement.builder()
    .attributeName("year")
    .keyType(KeyType.HASH)
    .build();

KeySchemaElement key2 = KeySchemaElement.builder()
    .attributeName("title")
    .keyType(KeyType.RANGE)
    .build();

// Add KeySchemaElement objects to the list.
tableKey.add(key);
tableKey.add(key2);

CreateTableRequest request = CreateTableRequest.builder()
    .keySchema(tableKey)
    .provisionedThroughput(ProvisionedThroughput.builder()
        .readCapacityUnits(10L)
        .writeCapacityUnits(10L)
        .build())
    .attributeDefinitions(attributeDefinitions)
    .tableName(tableName)
    .build();

try {
    CreateTableResponse response = ddb.createTable(request);
    DescribeTableRequest tableRequest = DescribeTableRequest.builder()
        .tableName(tableName)
        .build();

    // Wait until the Amazon DynamoDB table is created.
    WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    String newTable = response.tableDescription().tableName();
    System.out.println("The " + newTable + " was successfully created.");

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
}
```

```
        System.exit(1);
    }
}

// Query the table.
public static void queryTable(DynamoDbClient ddb) {
    try {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
        QueryConditional queryConditional = QueryConditional
            .keyEqualTo(Key.builder()
                .partitionValue(2013)
                .build());

        // Get items in the table and write out the ID value.
        Iterator<Movies> results =
custTable.query(queryConditional).items().iterator();
        String result = "";

        while (results.hasNext()) {
            Movies rec = results.next();
            System.out.println("The title of the movie is " + rec.getTitle());
            System.out.println("The movie information is " + rec.getInfo());
        }

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Scan the table.
public static void scanMovies(DynamoDbClient ddb, String tableName) {
    System.out.println("***** Scanning all movies.\n");
    try {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();
    }
}
```

```
DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
Iterator<Movies> results = custTable.scan().items().iterator();
while (results.hasNext()) {
    Movies rec = results.next();
    System.out.println("The movie title is " + rec.getTitle());
    System.out.println("The movie year is " + rec.getYear());
}

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}

// Load data into the table.
public static void loadData(DynamoDbClient ddb, String tableName, String
fileName) throws IOException {
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(ddb)
        .build();

    DynamoDbTable<Movies> mappedTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0;
    while (iter.hasNext()) {
        // Only add 200 Movies to the table.
        if (t == 200)
            break;
        currentNode = (ObjectNode) iter.next();

        int year = currentNode.path("year").asInt();
        String title = currentNode.path("title").asText();
        String info = currentNode.path("info").toString();

        Movies movies = new Movies();
        movies.setYear(year);
        movies.setTitle(title);
        movies.setInfo(info);
    }
}
```

```
        // Put the data into the Amazon DynamoDB Movie table.
        mappedTable.putItem(movies);
        t++;
    }
}

// Update the record to include show only directors.
public static void updateTableItem(DynamoDbClient ddb, String tableName) {
    HashMap<String, AttributeValue> itemKey = new HashMap<>();
    itemKey.put("year", AttributeValue.builder().n("1933").build());
    itemKey.put("title", AttributeValue.builder().s("King Kong").build());

    HashMap<String, AttributeValueUpdate> updatedValues = new HashMap<>();
    updatedValues.put("info", AttributeValueUpdate.builder()
        .value(AttributeValue.builder().s("{\"directors\":[\"Merian C.
Cooper\", \"Ernest B. Schoedsack\"]}")
        .build())
        .action(AttributeAction.PUT)
        .build());

    UpdateItemRequest request = UpdateItemRequest.builder()
        .tableName(tableName)
        .key(itemKey)
        .attributeUpdates(updatedValues)
        .build();

    try {
        ddb.updateItem(request);
    } catch (ResourceNotFoundException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    System.out.println("Item was updated!");
}

public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
    DeleteTableRequest request = DeleteTableRequest.builder()
        .tableName(tableName)
        .build();
```

```
    try {
        ddb.deleteTable(request);

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println(tableName + " was successfully deleted!");
}

public static void putRecord(DynamoDbClient ddb) {
    try {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> table = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));

        // Populate the Table.
        Movies record = new Movies();
        record.setYear(2020);
        record.setTitle("My Movie2");
        record.setInfo("no info");
        table.putItem(record);

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Added a new movie to the table.");
}

public static void getItem(DynamoDbClient ddb) {

    HashMap<String, AttributeValue> keyToGet = new HashMap<>();
    keyToGet.put("year", AttributeValue.builder()
        .n("1933")
        .build());

    keyToGet.put("title", AttributeValue.builder()
        .s("King Kong")
        .build());
```



```
GetItemRequest request = GetItemRequest.builder()
    .key(keyToGet)
    .tableName("Movies")
    .build();

try {
    Map<String, AttributeValue> returnedItem = ddb.getItem(request).item();

    if (returnedItem != null) {
        Set<String> keys = returnedItem.keySet();
        System.out.println("Amazon DynamoDB table attributes: \n");

        for (String key1 : keys) {
            System.out.format("%s: %s\n", key1,
returnedItem.get(key1).toString());
        }
    } else {
        System.out.format("No item found with the key %s!\n", "year");
    }

} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [BatchWriteItem](#)
 - [CreateTable](#)
 - [DeleteItem](#)
 - [DeleteTable](#)
 - [DescribeTable](#)
 - [GetItem](#)
 - [PutItem](#)
 - [查询](#)
 - [扫描](#)

- [UpdateItem](#)

使用批量 PartiQL 语句查询表

以下代码示例展示了如何：

- 通过运行多个SELECT语句获取一批项目。
- 通过运行多个INSERT语句来添加一批项目。
- 通过运行多个UPDATE语句来更新一批项目。
- 通过运行多个DELETE语句来删除一批项目。

SDK适用于 Java 2.x

Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public class ScenarioPartiQLBatch {
    public static void main(String[] args) throws IOException {
        String tableName = "MoviesPartiQLBatch";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        System.out.println("***** Creating an Amazon DynamoDB table named
" + tableName
            + " with a key named year and a sort key named
title.");
        createTable(ddb, tableName);

        System.out.println("***** Adding multiple records into the " +
tableName
            + " table using a batch command.");
        putRecordBatch(ddb);
    }
}
```

```
        System.out.println("***** Updating multiple records using a batch
command.");
        updateTableItemBatch(ddb);

        System.out.println("***** Deleting multiple records using a batch
command.");
        deleteItemBatch(ddb);

        System.out.println("***** Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new
ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE) // Sort
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);
    }
}
```

```

        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)

        .provisionedThroughput(ProvisionedThroughput.builder()
            .readCapacityUnits(new Long(10))
            .writeCapacityUnits(new Long(10))
            .build())
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest =
DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter
                .waitUntilTableExists(tableRequest);

            waiterResponse.matched().response().ifPresent(System.out::println);
            String newTable = response.tableDescription().tableName();
            System.out.println("The " + newTable + " was successfully
created.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void putRecordBatch(DynamoDbClient ddb) {
        String sqlStatement = "INSERT INTO MoviesPartiQBatch VALUE
{'year':?, 'title' : ?, 'info' : ?}";
        try {
            // Create three movies to add to the Amazon DynamoDB table.
            // Set data for Movie 1.
            List<AttributeValue> parameters = new ArrayList<>();

            AttributeValue att1 = AttributeValue.builder()
                .n(String.valueOf("2022"))

```

```
        .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("My Movie 1")
            .build();

        AttributeValue att3 = AttributeValue.builder()
            .s("No Information")
            .build();

        parameters.add(att1);
        parameters.add(att2);
        parameters.add(att3);

        BatchStatementRequest statementRequestMovie1 =
BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parameters)
            .build();

        // Set data for Movie 2.
        List<AttributeValue> parametersMovie2 = new ArrayList<>();
        AttributeValue attMovie2 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attMovie2A = AttributeValue.builder()
            .s("My Movie 2")
            .build();

        AttributeValue attMovie2B = AttributeValue.builder()
            .s("No Information")
            .build();

        parametersMovie2.add(attMovie2);
        parametersMovie2.add(attMovie2A);
        parametersMovie2.add(attMovie2B);

        BatchStatementRequest statementRequestMovie2 =
BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersMovie2)
            .build();
```

```
// Set data for Movie 3.
List<AttributeValue> parametersMovie3 = new ArrayList<>();
AttributeValue attMovie3 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attMovie3A = AttributeValue.builder()
    .s("My Movie 3")
    .build();

AttributeValue attMovie3B = AttributeValue.builder()
    .s("No Information")
    .build();

parametersMovie3.add(attMovie3);
parametersMovie3.add(attMovie3A);
parametersMovie3.add(attMovie3B);

BatchStatementRequest statementRequestMovie3 =
BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parametersMovie3)
    .build();

// Add all three movies to the list.
List<BatchStatementRequest> myBatchStatementList = new
ArrayList<>();

myBatchStatementList.add(statementRequestMovie1);
myBatchStatementList.add(statementRequestMovie2);
myBatchStatementList.add(statementRequestMovie3);

BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
    .statements(myBatchStatementList)
    .build();

BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchRequest);
System.out.println("ExecuteStatement successful: " +
response.toString());
System.out.println("Added new movies using a batch
command.");

} catch (DynamoDbException e) {
```

```
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateTableItemBatch(DynamoDbClient ddb) {
    String sqlStatement = "UPDATE MoviesPartiQBatch SET info =
'directors\":[\"Merian C. Cooper\", \"Ernest B. Schoedsack' where year=? and
title=?";
    List<AttributeValue> parametersRec1 = new ArrayList<>();

    // Update three records.
    AttributeValue att1 = AttributeValue.builder()
        .n(String.valueOf("2022"))
        .build();

    AttributeValue att2 = AttributeValue.builder()
        .s("My Movie 1")
        .build();

    parametersRec1.add(att1);
    parametersRec1.add(att2);

    BatchStatementRequest statementRequestRec1 =
BatchStatementRequest.builder()
        .statement(sqlStatement)
        .parameters(parametersRec1)
        .build();

    // Update record 2.
    List<AttributeValue> parametersRec2 = new ArrayList<>();
    AttributeValue attRec2 = AttributeValue.builder()
        .n(String.valueOf("2022"))
        .build();

    AttributeValue attRec2a = AttributeValue.builder()
        .s("My Movie 2")
        .build();

    parametersRec2.add(attRec2);
    parametersRec2.add(attRec2a);
    BatchStatementRequest statementRequestRec2 =
BatchStatementRequest.builder()
        .statement(sqlStatement)
```

```
        .parameters(parametersRec2)
        .build();

// Update record 3.
List<AttributeValue> parametersRec3 = new ArrayList<>();
AttributeValue attRec3 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attRec3a = AttributeValue.builder()
    .s("My Movie 3")
    .build();

parametersRec3.add(attRec3);
parametersRec3.add(attRec3a);
BatchStatementRequest statementRequestRec3 =
BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parametersRec3)
    .build();

// Add all three movies to the list.
List<BatchStatementRequest> myBatchStatementList = new
ArrayList<>();
myBatchStatementList.add(statementRequestRec1);
myBatchStatementList.add(statementRequestRec2);
myBatchStatementList.add(statementRequestRec3);

BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
    .statements(myBatchStatementList)
    .build();

try {
    BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchRequest);
    System.out.println("ExecuteStatement successful: " +
response.toString());
    System.out.println("Updated three movies using a batch
command.");
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```



```
        }
        System.out.println("Item was updated!");
    }

    public static void deleteItemBatch(DynamoDbClient ddb) {
        String sqlStatement = "DELETE FROM MoviesPartiQBatch WHERE year = ?
and title=?";
        List<AttributeValue> parametersRec1 = new ArrayList<>();

        // Specify three records to delete.
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("My Movie 1")
            .build();

        parametersRec1.add(att1);
        parametersRec1.add(att2);

        BatchStatementRequest statementRequestRec1 =
BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec1)
            .build();

        // Specify record 2.
        List<AttributeValue> parametersRec2 = new ArrayList<>();
        AttributeValue attRec2 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attRec2a = AttributeValue.builder()
            .s("My Movie 2")
            .build();

        parametersRec2.add(attRec2);
        parametersRec2.add(attRec2a);
        BatchStatementRequest statementRequestRec2 =
BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec2)
            .build();
    }
}
```

```
// Specify record 3.
List<AttributeValue> parametersRec3 = new ArrayList<>();
AttributeValue attRec3 = AttributeValue.builder()
    .n(String.valueOf("2022"))
    .build();

AttributeValue attRec3a = AttributeValue.builder()
    .s("My Movie 3")
    .build();

parametersRec3.add(attRec3);
parametersRec3.add(attRec3a);

BatchStatementRequest statementRequestRec3 =
BatchStatementRequest.builder()
    .statement(sqlStatement)
    .parameters(parametersRec3)
    .build();

// Add all three movies to the list.
List<BatchStatementRequest> myBatchStatementList = new
ArrayList<>();
myBatchStatementList.add(statementRequestRec1);
myBatchStatementList.add(statementRequestRec2);
myBatchStatementList.add(statementRequestRec3);

BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
    .statements(myBatchStatementList)
    .build();

try {
    ddb.batchExecuteStatement(batchRequest);
    System.out.println("Deleted three movies using a batch
command.");
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

```
public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName)
{
    DeleteTableRequest request = DeleteTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        ddb.deleteTable(request);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println(tableName + " was successfully deleted!");
}

private static ExecuteStatementResponse
executeStatementRequest(DynamoDbClient ddb, String statement,
    List<AttributeValue> parameters) {
    ExecuteStatementRequest request = ExecuteStatementRequest.builder()
        .statement(statement)
        .parameters(parameters)
        .build();

    return ddb.executeStatement(request);
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [BatchExecuteStatement](#)”中的。

使用 PartiQL 来查询表

以下代码示例展示了如何：

- 通过运行SELECT语句获取项目。
- 通过运行INSERT语句添加项目。
- 通过运行UPDATE语句更新项目。
- 通过运行DELETE语句删除项目。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public class ScenarioPartiQ {
    public static void main(String[] args) throws IOException {
        final String usage = ""

                Usage:
                <fileName>

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

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String fileName = args[0];
        String tableName = "MoviesPartiQ";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        System.out.println(
            "***** Creating an Amazon DynamoDB table named MoviesPartiQ with a
key named year and a sort key named title.");
        createTable(ddb, tableName);

        System.out.println("***** Loading data into the MoviesPartiQ table.");
        loadData(ddb, fileName);

        System.out.println("***** Getting data from the MoviesPartiQ table.");
        getItem(ddb);
    }
}
```

```
        System.out.println("***** Putting a record into the MoviesPartiQ table.");
        putRecord(ddb);

        System.out.println("***** Updating a record.");
        updateTableItem(ddb);

        System.out.println("***** Querying the movies released in 2013.");
        queryTable(ddb);

        System.out.println("***** Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE) // Sort
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);
    }
}
```

```
        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)
            .provisionedThroughput(ProvisionedThroughput.builder()
                .readCapacityUnits(new Long(10))
                .writeCapacityUnits(new Long(10))
                .build())
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            String newTable = response.tableDescription().tableName();
            System.out.println("The " + newTable + " was successfully created.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Load data into the table.
    public static void loadData(DynamoDbClient ddb, String fileName) throws
IOException {

        String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?,
'title' : ?, 'info' : ?}";
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0;
        List<AttributeValue> parameters = new ArrayList<>();
        while (iter.hasNext()) {
```

```
// Add 200 movies to the table.
if (t == 200)
    break;
currentNode = (ObjectNode) iter.next();

int year = currentNode.path("year").asInt();
String title = currentNode.path("title").asText();
String info = currentNode.path("info").toString();

AttributeValue att1 = AttributeValue.builder()
    .n(String.valueOf(year))
    .build();

AttributeValue att2 = AttributeValue.builder()
    .s(title)
    .build();

AttributeValue att3 = AttributeValue.builder()
    .s(info)
    .build();

parameters.add(att1);
parameters.add(att2);
parameters.add(att3);

// Insert the movie into the Amazon DynamoDB table.
executeStatementRequest(ddb, sqlStatement, parameters);
System.out.println("Added Movie " + title);

parameters.remove(att1);
parameters.remove(att2);
parameters.remove(att3);
t++;
}
}

public static void getItem(DynamoDbClient ddb) {

    String sqlStatement = "SELECT * FROM MoviesPartiQ where year=? and title=?";
    List<AttributeValue> parameters = new ArrayList<>();
    AttributeValue att1 = AttributeValue.builder()
        .n("2012")
        .build();
```

```
        AttributeValue att2 = AttributeValue.builder()
            .s("The Perks of Being a Wallflower")
            .build();

        parameters.add(att1);
        parameters.add(att2);

        try {
            ExecuteStatementResponse response = executeStatementRequest(ddb,
                sqlStatement, parameters);
            System.out.println("ExecuteStatement successful: " +
                response.toString());
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void putRecord(DynamoDbClient ddb) {

        String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?,
            'title' : ?, 'info' : ?}";
        try {
            List<AttributeValue> parameters = new ArrayList<>();

            AttributeValue att1 = AttributeValue.builder()
                .n(String.valueOf("2020"))
                .build();

            AttributeValue att2 = AttributeValue.builder()
                .s("My Movie")
                .build();

            AttributeValue att3 = AttributeValue.builder()
                .s("No Information")
                .build();

            parameters.add(att1);
            parameters.add(att2);
            parameters.add(att3);

            executeStatementRequest(ddb, sqlStatement, parameters);
        }
    }
}
```



```
        System.out.println("Added new movie.");

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateTableItem(DynamoDbClient ddb) {

    String sqlStatement = "UPDATE MoviesPartiQ SET info = 'directors\":[\"Merian
C. Cooper\", \"Ernest B. Schoedsack' where year=? and title=?";
    List<AttributeValue> parameters = new ArrayList<>();
    AttributeValue att1 = AttributeValue.builder()
        .n(String.valueOf("2013"))
        .build();

    AttributeValue att2 = AttributeValue.builder()
        .s("The East")
        .build();

    parameters.add(att1);
    parameters.add(att2);

    try {
        executeStatementRequest(ddb, sqlStatement, parameters);

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("Item was updated!");
}

// Query the table where the year is 2013.
public static void queryTable(DynamoDbClient ddb) {
    String sqlStatement = "SELECT * FROM MoviesPartiQ where year = ? ORDER BY
year";
    try {

        List<AttributeValue> parameters = new ArrayList<>();
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2013"))
            .build();
```

```
        parameters.add(att1);

        // Get items in the table and write out the ID value.
        ExecuteStatementResponse response = executeStatementRequest(ddb,
sqlStatement, parameters);
        System.out.println("ExecuteStatement successful: " +
response.toString());

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {

    DeleteTableRequest request = DeleteTableRequest.builder()
        .tableName(tableName)
        .build();

    try {
        ddb.deleteTable(request);

    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    System.out.println(tableName + " was successfully deleted!");
}

private static ExecuteStatementResponse executeStatementRequest(DynamoDbClient
ddb, String statement,
    List<AttributeValue> parameters) {
    ExecuteStatementRequest request = ExecuteStatementRequest.builder()
        .statement(statement)
        .parameters(parameters)
        .build();

    return ddb.executeStatement(request);
}

private static void processResults(ExecuteStatementResponse
executeStatementResult) {
```

```
        System.out.println("ExecuteStatement successful: " +
            executeStatementResult.toString());
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ExecuteStatement](#)”中的。

查询物TTL品

以下代码示例显示了如何查询TTL项目。

SDK适用于 Java 2.x

查询筛选表达式以收集 DynamoDB 表中的TTL项目。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.utils.ImmutableMap;

import java.util.Map;
import java.util.Optional;

// Get current time in epoch second format (comparing against expiry
attribute)
final long currentTime = System.currentTimeMillis() / 1000;

// A string that contains conditions that DynamoDB applies after the Query
operation, but before the data is returned to you.
final String keyConditionExpression = "#pk = :pk";

// The condition that specifies the key values for items to be retrieved by
the Query action.
final String filterExpression = "#ea > :ea";
final Map<String, String> expressionAttributeNames = ImmutableMap.of(
    "#pk", "primaryKey",
    "#ea", "expireAt");
```

```
final Map<String, AttributeValue> expressionAttributeValues =
ImmutableMap.of(
    ":pk", AttributeValue.builder().s(primaryKey).build(),
    ":ea",
AttributeValue.builder().s(String.valueOf(currentTime)).build()
);

final QueryRequest request = QueryRequest.builder()
    .tableName(tableName)
    .keyConditionExpression(keyConditionExpression)
    .filterExpression(filterExpression)
    .expressionAttributeNames(expressionAttributeNames)
    .expressionAttributeValues(expressionAttributeValues)
    .build();
try (DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build()) {
    final QueryResponse response = ddb.query(request);
    System.out.println(tableName + " Query operation with TTL successful.
Request id is "
        + response.responseMetadata().requestId());
    // Print the items that are not expired
    for (Map<String, AttributeValue> item : response.items()) {
        System.out.println(item.toString());
    }
} catch (ResourceNotFoundException e) {
    System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
    System.exit(1);
} catch (DynamoDbException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
System.exit(0);
```

- 有关API详细信息，请参阅AWS SDK for Java 2.x API参考文献中的[查询](#)。

更新物品的 TTL

以下代码示例显示了如何更新项目TTL。

SDK适用于 Java 2.x

更新表TTL中现有 DynamoDB 项目。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;
import software.amazon.awssdk.utils.ImmutableMap;

import java.util.Map;
import java.util.Optional;

// Get current time in epoch second format
final long currentTime = System.currentTimeMillis() / 1000;
// Calculate expiration time 90 days from now in epoch second format
final long expireDate = currentTime + (90 * 24 * 60 * 60);
// An expression that defines one or more attributes to be updated, the
action to be performed on them, and new values for them.
final String updateExpression = "SET updatedAt=:c, expireAt=:e";

final ImmutableMap<String, AttributeValue> keyMap =
    ImmutableMap.of("primaryKey", AttributeValue.fromS(primaryKey),
        "sortKey", AttributeValue.fromS(sortKey));
final Map<String, AttributeValue> expressionAttributeValues =
ImmutableMap.of(
    ":c",
AttributeValue.builder().s(String.valueOf(currentTime)).build(),
    ":e", AttributeValue.builder().s(String.valueOf(expireDate)).build()
);

final UpdateItemRequest request = UpdateItemRequest.builder()
    .tableName(tableName)
    .key(keyMap)
    .updateExpression(updateExpression)
    .expressionAttributeValues(expressionAttributeValues)
    .build();
try (DynamoDbClient ddb = DynamoDbClient.builder()
    .region(region)
    .build()) {
    final UpdateItemResponse response = ddb.updateItem(request);
```

```
        System.out.println(tableName + " UpdateItem operation with TTL
successful. Request id is "
            + response.responseMetadata().requestId());
    } catch (ResourceNotFoundException e) {
        System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
        System.exit(1);
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.exit(0);
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [UpdateItem](#)”中的。

无服务器示例

通过 DynamoDB 触发器调用 Lambda 函数

以下代码示例演示如何实现 Lambda 函数，该函数接收通过从 DynamoDB 流接收记录而触发的事件。该函数检索 DynamoDB 有效负载，并记录下记录内容。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

使用 Java 将 DynamoDB 事件与 Lambda 结合使用。

```
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()));
    }
}
```

通过 DynamoDB 触发器报告 Lambda 函数批处理项目失败

以下代码示例演示如何为接收来自 DynamoDB 流的事件的 Lambda 函数实现部分批量响应。该函数在响应中报告批处理项目失败，并指示 Lambda 稍后重试这些消息。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

报告使用 Java 通过 Lambda 进行 DynamoDB 批处理项目失败。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;
import com.amazonaws.services.lambda.runtime.events.models.dynamodb.StreamRecord;
```

```
import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class ProcessDynamodbRecords implements RequestHandler<DynamodbEvent,
    Serializable> {

    @Override
    public StreamsEventResponse handleRequest(DynamodbEvent input, Context context)
    {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
        ArrayList<>();
        String curRecordSequenceNumber = "";

        for (DynamodbEvent.DynamodbStreamRecord dynamodbStreamRecord :
        input.getRecords()) {
            try {
                //Process your record
                StreamRecord dynamodbRecord = dynamodbStreamRecord.getDynamodb();
                curRecordSequenceNumber = dynamodbRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed item
                immediately.
                Lambda will immediately begin to retry processing from this
                failed item onwards. */
                batchItemFailures.add(new
                StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

        return new StreamsEventResponse();
    }
}
```

使用SDK适用于 Java 2.x 的亚马逊EC2示例

以下代码示例向您展示如何在 Amazon 中使用来执行操作和实现常见场景EC2。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。


每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

开始使用

你好 Amazon EC2

以下代码示例展示了如何开始使用 Amazon EC2。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
    try {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
            .groupIds(groupId)
            .build();

        // Use a paginator.
        DescribeSecurityGroupsIterable listGroups =
ec2.describeSecurityGroupsPaginator(request);
        listGroups.stream()
            .flatMap(r -> r.securityGroups().stream())
            .forEach(group -> System.out
                .println(" Group id: " +group.groupId() + " group name = " +
group.groupName()));
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeSecurityGroups](#)”中的。

主题

- [操作](#)
- [场景](#)

操作

AllocateAddress

以下代码示例演示如何使用 AllocateAddress。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static String allocateAddress(Ec2Client ec2) {
    try {
        AllocateAddressRequest allocateRequest =
        AllocateAddressRequest.builder()
            .domain(DomainType.VPC)
            .build();

        AllocateAddressResponse allocateResponse =
        ec2.allocateAddress(allocateRequest);
        return allocateResponse.allocationId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

```
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [AllocateAddress](#)”中的。

AssociateAddress

以下代码示例演示如何使用 AssociateAddress。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static String associateAddress(Ec2Client ec2, String instanceId, String
allocationId) {
    try {
        AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
            .instanceId(instanceId)
            .allocationId(allocationId)
            .build();

        AssociateAddressResponse associateResponse =
ec2.associateAddress(associateRequest);
        return associateResponse.associationId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [AssociateAddress](#)”中的。

AuthorizeSecurityGroupIngress

以下代码示例演示如何使用 AuthorizeSecurityGroupIngress。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static String createSecurityGroup(Ec2Client ec2, String groupName, String
groupDesc, String vpcId,
    String myIpAddress) {
    try {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

        CreateSecurityGroupResponse resp =
ec2.createSecurityGroup(createRequest);
        IpRange ipRange = IpRange.builder()
            .cidrIp(myIpAddress + "/0")
            .build();

        IpPermission ipPerm = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(80)
            .fromPort(80)
            .ipRanges(ipRange)
            .build();

        IpPermission ipPerm2 = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(22)
            .fromPort(22)
            .ipRanges(ipRange)
            .build();
```

```
        AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
    .groupName(groupName)
    .ipPermissions(ipPerm, ipPerm2)
    .build();

    ec2.authorizeSecurityGroupIngress(authRequest);
    System.out.println("Successfully added ingress policy to security group
" + groupName);
    return resp.groupId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [AuthorizeSecurityGroupIngress](#)”中的。

CreateKeyPair

以下代码示例演示如何使用 CreateKeyPair。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void createKeyPair(Ec2Client ec2, String keyName, String fileName)
{
    try {
        CreateKeyPairRequest request = CreateKeyPairRequest.builder()
            .keyName(keyName)
            .build();
```

```
        CreateKeyPairResponse response = ec2.createKeyPair(request);
        String content = response.keyMaterial();
        BufferedWriter writer = new BufferedWriter(new FileWriter(fileName));
        writer.write(content);
        writer.close();
        System.out.println("Successfully created key pair named " + keyName);

    } catch (Ec2Exception | IOException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateKeyPair](#)”中的。

CreateSecurityGroup

以下代码示例演示如何使用 CreateSecurityGroup。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static String createSecurityGroup(Ec2Client ec2, String groupName, String
groupDesc, String vpcId,
    String myIpAddress) {
    try {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

        CreateSecurityGroupResponse resp =
ec2.createSecurityGroup(createRequest);
```

```
        IpRange ipRange = IpRange.builder()
            .cidrIp(myIpAddress + "/0")
            .build();

        IpPermission ipPerm = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(80)
            .fromPort(80)
            .ipRanges(ipRange)
            .build();

        IpPermission ipPerm2 = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(22)
            .fromPort(22)
            .ipRanges(ipRange)
            .build();

        AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(groupName)
            .ipPermissions(ipPerm, ipPerm2)
            .build();

        ec2.authorizeSecurityGroupIngress(authRequest);
        System.out.println("Successfully added ingress policy to security group
" + groupName);
        return resp.groupId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateSecurityGroup](#)”中的。

DeleteKeyPair

以下代码示例演示如何使用 DeleteKeyPair。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void deleteKeys(Ec2Client ec2, String keyPair) {
    try {
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
            .keyName(keyPair)
            .build();

        ec2.deleteKeyPair(request);
        System.out.println("Successfully deleted key pair named " + keyPair);
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteKeyPair](#)”中的。

DeleteSecurityGroup

以下代码示例演示如何使用 DeleteSecurityGroup。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void deleteEC2SecGroup(Ec2Client ec2, String groupId) {
    try {
```



```
        DeleteSecurityGroupRequest request =
DeleteSecurityGroupRequest.builder()
        .groupId(groupId)
        .build();

        ec2.deleteSecurityGroup(request);
        System.out.println("Successfully deleted security group with Id " +
groupId);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteSecurityGroup](#)”中的。

DescribeInstanceTypes

以下代码示例演示如何使用 DescribeInstanceTypes。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
// Get a list of instance types.
public static String getInstanceTypes(Ec2Client ec2) {
    String instanceType;
    try {
        DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
        .maxResults(10)
        .build();

        DescribeInstanceTypesResponse response =
ec2.describeInstanceTypes(typesRequest);
        List<InstanceTypeInfo> instanceTypes = response.instanceTypes();
```

```
        for (InstanceTypeInfo type : instanceTypes) {
            System.out.println("The memory information of this type is " +
                type.memoryInfo().sizeInMiB());
            System.out.println("Network information is " +
                type.networkInfo().toString());
            System.out.println("Instance type is " +
                type.instanceType().toString());
            instanceType = type.instanceType().toString();
            if (instanceType.compareTo("t2.2xlarge") == 0){
                return instanceType;
            }
        }

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeInstanceTypes](#)”中的。

DescribeInstances

以下代码示例演示如何使用 DescribeInstances。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.paginators.DescribeInstancesIterable;

/**
```

```
* Before running this Java V2 code example, set up your development
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DescribeInstances {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        Ec2Client ec2 = Ec2Client.builder()
            .region(region)
            .build();

        describeEC2Instances(ec2);
        ec2.close();
    }

    public static void describeEC2Instances(Ec2Client ec2) {
        try {
            DescribeInstancesRequest request = DescribeInstancesRequest.builder()
                .maxResults(10)
                .build();

            DescribeInstancesIterable instancesIterable =
ec2.describeInstancesPaginator(request);
            instancesIterable.stream()
                .flatMap(r -> r.reservations().stream())
                .flatMap(reservation -> reservation.instances().stream())
                .forEach(instance -> {
                    System.out.println("Instance Id is " + instance.instanceId());
                    System.out.println("Image id is " + instance.imageId());
                    System.out.println("Instance type is " +
instance.instanceType());
                    System.out.println("Instance state name is " +
instance.state().name());
                    System.out.println("Monitoring information is " +
instance.monitoring().state());
                });

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorCode());
            System.exit(1);
        }
    }
}
```

```
}  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeInstances](#)”中的。

DescribeKeyPairs

以下代码示例演示如何使用 DescribeKeyPairs。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void describeKeys(Ec2Client ec2) {  
    try {  
        DescribeKeyPairsResponse response = ec2.describeKeyPairs();  
        response.keyPairs().forEach(keyPair -> System.out.printf(  
            "Found key pair with name %s " +  
            "and fingerprint %s",  
            keyPair.keyName(),  
            keyPair.keyFingerprint()));  
    } catch (Ec2Exception e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeKeyPairs](#)”中的。

DescribeSecurityGroups

以下代码示例演示如何使用 DescribeSecurityGroups。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
    try {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
            .groupIds(groupId)
            .build();

        // Use a paginator.
        DescribeSecurityGroupsIterable listGroups =
ec2.describeSecurityGroupsPaginator(request);
        listGroups.stream()
            .flatMap(r -> r.securityGroups().stream())
            .forEach(group -> System.out
                .println(" Group id: " +group.groupId() + " group name = " +
group.groupName()));
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeSecurityGroups](#)”中的。

DisassociateAddress

以下代码示例演示如何使用 DisassociateAddress。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void disassociateAddress(Ec2Client ec2, String associationId) {
    try {
        DisassociateAddressRequest addressRequest =
        DisassociateAddressRequest.builder()
            .associationId(associationId)
            .build();

        ec2.disassociateAddress(addressRequest);
        System.out.println("You successfully disassociated the address!");
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DisassociateAddress](#)”中的。

GetPasswordData

以下代码示例演示如何使用 GetPasswordData。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.core.exception.SdkClientException;
```

```
import software.amazon.awssdk.core.exception.SdkServiceException;
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.GetPasswordDataRequest;
import software.amazon.awssdk.services.ec2.model.GetPasswordDataResponse;
import
    software.amazon.awssdk.services.secretsmanager.model.ResourceNotFoundException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * 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 from which the password is obtained.
\s
        """;

        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        Region region = Region.US_EAST_1;
        Ec2Client ec2 = Ec2Client.builder()
            .region(region)
            .build();

        String instanceId = args[0];
        getPasswordData(ec2,instanceId);
    }
}
```

```
/**
 * Retrieves and prints the encrypted administrator password data for a
 * specified EC2 instance.
 *
 * <p>The password data is encrypted using the key pair that was specified when
 * the instance was launched.
 * To decrypt the password data, you can use the private key of the key pair.</
 * p>
 *
 * @param ec2      The {@link Ec2Client} to use for making the request.
 * @param instanceId The ID of the instance for which to get the encrypted
 * password data.
 */
public static void getPasswordData(Ec2Client ec2, String instanceId) {
    GetPasswordDataRequest getPasswordDataRequest =
    GetPasswordDataRequest.builder()
        .instanceId(instanceId)
        .build();

    try {
        GetPasswordDataResponse getPasswordDataResponse =
        ec2.getPasswordData(getPasswordDataRequest);
        String encryptedPasswordData = getPasswordDataResponse.passwordData();
        System.out.println("Encrypted Password Data: " + encryptedPasswordData);

    } catch (Ec2Exception e) {
        String errorCode = e.awsErrorDetails().errorCode();
        if (errorCode.matches("InvalidInstanceID.NotFound")) {
            System.err.println("Instance ID not found, unable to retrieve password
            data.");
        } else {
            System.err.println("There was a problem retrieving password data.
            Details:");
            e.printStackTrace();
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetPasswordData](#)”中的。

ReleaseAddress

以下代码示例演示如何使用 ReleaseAddress。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void releaseEC2Address(Ec2Client ec2, String allocId) {
    try {
        ReleaseAddressRequest request = ReleaseAddressRequest.builder()
            .allocationId(allocId)
            .build();

        ec2.releaseAddress(request);
        System.out.println("Successfully released Elastic IP address " +
allocId);
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ReleaseAddress](#)”中的。

RunInstances

以下代码示例演示如何使用 RunInstances。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.InstanceType;
import software.amazon.awssdk.services.ec2.model.RunInstancesRequest;
import software.amazon.awssdk.services.ec2.model.RunInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Tag;
import software.amazon.awssdk.services.ec2.model.CreateTagsRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This code example requires an AMI value. You can learn more about this value
 * by reading this documentation topic:
 *
 * https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/AMIs.html
 */
public class CreateInstance {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <name> <amiId>

                Where:
                name - An instance name value that you can obtain from the AWS
Console (for example, ami-xxxxxx5c8b987b1a0).\s
                amiId - An Amazon Machine Image (AMI) value that you can obtain
from the AWS Console (for example, i-xxxxxx2734106d0ab).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String name = args[0];
        String amiId = args[1];
```

```
Region region = Region.US_EAST_1;
Ec2Client ec2 = Ec2Client.builder()
    .region(region)
    .build();

String instanceId = createEC2Instance(ec2, name, amiId);
System.out.println("The Amazon EC2 Instance ID is " + instanceId);
ec2.close();
}

public static String createEC2Instance(Ec2Client ec2, String name, String amiId)
{
    RunInstancesRequest runRequest = RunInstancesRequest.builder()
        .imageId(amiId)
        .instanceType(InstanceType.T1_MICRO)
        .maxCount(1)
        .minCount(1)
        .build();

    // Use a waiter to wait until the instance is running.
    System.out.println("Going to start an EC2 instance using a waiter");
    RunInstancesResponse response = ec2.runInstances(runRequest);
    String instanceIdVal = response.instances().get(0).instanceId();
    ec2.waiter().waitUntilInstanceRunning(r -> r.instanceIds(instanceIdVal));
    Tag tag = Tag.builder()
        .key("Name")
        .value(name)
        .build();

    CreateTagsRequest tagRequest = CreateTagsRequest.builder()
        .resources(instanceIdVal)
        .tags(tag)
        .build();

    try {
        ec2.createTags(tagRequest);
        System.out.printf("Successfully started EC2 Instance %s based on AMI
%s", instanceIdVal, amiId);
        return instanceIdVal;
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
        return "";  
    }  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [RunInstances](#)”中的。

StartInstances

以下代码示例演示如何使用 StartInstances。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void startInstance(Ec2Client ec2, String instanceId) {  
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()  
        .overrideConfiguration(b -> b.maxAttempts(100))  
        .client(ec2)  
        .build();  
  
    StartInstancesRequest request = StartInstancesRequest.builder()  
        .instanceIds(instanceId)  
        .build();  
  
    System.out.println("Use an Ec2Waiter to wait for the instance to run. This  
will take a few minutes.");  
    ec2.startInstances(request);  
    DescribeInstancesRequest instanceRequest =  
DescribeInstancesRequest.builder()  
        .instanceIds(instanceId)  
        .build();  
  
    WaiterResponse<DescribeInstancesResponse> waiterResponse =  
ec2Waiter.waitUntilInstanceRunning(instanceRequest);  
    waiterResponse.matched().response().ifPresent(System.out::println);  
}
```

```
        System.out.println("Successfully started instance " + instanceId);
    }
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartInstances](#)”中的。

StopInstances

以下代码示例演示如何使用 StopInstances。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void stopInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();
    StopInstancesRequest request = StopInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to stop. This
will take a few minutes.");
    ec2.stopInstances(request);
    DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceStopped(instanceRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Successfully stopped instance " + instanceId);
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StopInstances](#)”中的。

TerminateInstances

以下代码示例演示如何使用 TerminateInstances。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void terminateEC2(Ec2Client ec2, String instanceId) {
    try {
        Ec2Waiter ec2Waiter = Ec2Waiter.builder()
            .overrideConfiguration(b -> b.maxAttempts(100))
            .client(ec2)
            .build();

        TerminateInstancesRequest ti = TerminateInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        System.out.println("Use an Ec2Waiter to wait for the instance to
        terminate. This will take a few minutes.");
        ec2.terminateInstances(ti);
        DescribeInstancesRequest instanceRequest =
        DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        WaiterResponse<DescribeInstancesResponse> waiterResponse = ec2Waiter
            .waitUntilInstanceTerminated(instanceRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Successfully started instance " + instanceId);
        System.out.println(instanceId + " is terminated!");
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [TerminateInstances](#)”中的。

场景

构建和管理弹性服务

以下代码示例演示了如何创建可返回书籍、电影和歌曲推荐的负载均衡的 Web 服务。该示例演示服务如何响应故障，以及如何重组服务以提高故障发生时的弹性。

- 使用 Amazon A EC2 uto Scaling 组根据启动模板创建亚马逊弹性计算云 (AmazonEC2) 实例，并将实例数量保持在指定范围内。
- 使用 Elastic Load Balancing 处理和分发HTTP请求。
- 监控自动扩缩组中实例的运行状况，并仅将请求转发到运行状况良好的实例。
- 在每个EC2实例上运行 Python 网络服务器来处理HTTP请求。Web 服务器以建议和运行状况检查作为响应。
- 使用 Amazon DynamoDB 表模拟推荐服务。
- 通过更新 AWS Systems Manager 参数来控制 Web 服务器对请求和运行状况检查的响应。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

在命令提示符中运行交互式场景。

```
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.deleteAutoScalingGroup(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 wasSuccessful = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
if (!wasSuccessful) {
    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 (wasSuccessful) {
    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 =
HttpClientClients.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);
}
}
```

创建一个包含 Auto Scaling 和 Amazon EC2 操作的类。

```
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 terminateInstanceRequest =
    TerminateInstanceInAutoScalingGroupRequest
        .builder()
        .instanceId(instanceId)
        .shouldDecrementDesiredCapacity(false)
        .build();

    getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceRequest);
    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.getInstanceProfile().getInstanceProfileName();
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.getAttachedPolicies();
for (AttachedPolicy attachedPolicy : attachedPolicies) {
    DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
        .roleName(roleName)
        .policyArn(attachedPolicy.getPolicyArn())
        .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.");
}
}

```

创建一个包含弹性负载均衡操作的类。

```
public class LoadBalancer {
```



```
public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

public ElasticLoadBalancingV2Client getLoadBalancerClient() {
    if (elasticLoadBalancingV2Client == null) {
        elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }

    return elasticLoadBalancingV2Client;
}

// Checks the health of the instances in the target group.
public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
    DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
        .names(targetGroupName)
        .build();

    DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

    DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
        .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
        .build();

    DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
    return healthResponse.targetHealthDescriptions();
}

// Gets the HTTP endpoint of the load balancer.
public String getEndpoint(String lbName) {
    DescribeLoadBalancersResponse res = getLoadBalancerClient()
        .describeLoadBalancers(describe -> describe.names(lbName));
    return res.loadBalancers().get(0).dnsName();
}

// Deletes a load balancer.
public void deleteLoadBalancer(String lbName) {
    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
```

```

        .describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
            .build();

        getLoadBalancerClient().deleteLoadBalancer(
            builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancersDeleted(request);
        waiterResponse.matched().response().ifPresent(System.out::println);

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}

// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
            .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
            .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}

// Verify this computer can successfully send a GET request to the load balancer
// endpoint.
public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
    boolean success = false;
    int retries = 3;
    CloseableHttpClient httpClient = HttpClients.createDefault();

```

```
// Create an HTTP GET request to the ELB.
HttpGet httpGet = new HttpGet("http://" + elbDnsName);
try {
    while ((!success) && (retries > 0)) {
        // Execute the request and get the response.
        HttpResponse response = httpClient.execute(httpGet);
        int statusCode = response.getStatusLine().getStatusCode();
        System.out.println("HTTP Status Code: " + statusCode);
        if (statusCode == 200) {
            success = true;
        } else {
            retries--;
            System.out.println("Got connection error from load balancer
endpoint, retrying...");
            TimeUnit.SECONDS.sleep(15);
        }
    }

    } catch (org.apache.http.conn.HttpHostConnectException e) {
        System.out.println(e.getMessage());
    }

    System.out.println("Status.." + success);
    return success;
}

/*
 * Creates an Elastic Load Balancing target group. The target group specifies
 * how
 * the load balancer forward requests to instances in the group and how instance
 * health is checked.
 */
public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
        .name(targetGroupName)
        .protocol(protocol)
        .build();
}
```

```
        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /**
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
        String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                .map(Subnet::subnetId)
                .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                .subnets(subnetIdStrings)
                .name(lbName)
                .scheme("internet-facing")
                .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                .loadBalancerArns(lbARN)
                .build();
```

```
        System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

        // Create a listener for the load balance.
        Action action = Action.builder()
            .targetGroupArn(targetGroupARN)
            .type("forward")
            .build();

        CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
            .defaultActions(action)
            .port(port)
            .protocol(protocol)
            .defaultActions(action)
            .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
            + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
}
```

创建一个使用 DynamoDB 模拟推荐服务的类。

```
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);
}
}
```


创建一个包含 Systems Manager 操作的类。

```
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);
    }
}
```

• 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。

- [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](#)

开始使用实例

以下代码示例演示了操作流程：

- 创建密钥对和安全组。
- 选择 Amazon 系统映像 (AMI) 和兼容的实例类型，然后创建实例。
- 停止实例，然后再重启。
- 将弹性 IP 地址与您的实例相关联。
- 使用连接您的实例SSH，然后清理资源。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Creates an RSA key pair and saves the private key data as a .pem file.
 * 2. Lists key pairs.
 * 3. Creates a security group for the default VPC.
 * 4. Displays security group information.
 * 5. Gets a list of Amazon Linux 2 AMIs and selects one.
 * 6. Gets more information about the image.
 * 7. Gets a list of instance types that are compatible with the selected AMI's
 * architecture.
 * 8. Creates an instance with the key pair, security group, AMI, and an
 * instance type.
 * 9. Displays information about the instance.
 * 10. Stops the instance and waits for it to stop.
 * 11. Starts the instance and waits for it to start.
 * 12. Allocates an Elastic IP address and associates it with the instance.
 * 13. Displays SSH connection info for the instance.
 * 14. Disassociates and deletes the Elastic IP address.
 * 15. Terminates the instance and waits for it to terminate.
 * 16. Deletes the security group.
 * 17. Deletes the key pair.
 */
public class EC2Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
```

```

final String usage = ""

    Usage:
        <keyName> <fileName> <groupName> <groupDesc> <vpcId>

    Where:
        keyName - A key pair name (for example, TestKeyPair).\s
        fileName - A file name where the key information is written to.
\s
        groupName - The name of the security group.\s
        groupDesc - The description of the security group.\s
        vpcId - A VPC Id value. You can get this value from the AWS
Management Console.\s
        myIpAddress - The IP address of your development machine.\s

    """;

if (args.length != 6) {
    System.out.println(usage);
    System.exit(1);
}

String keyName = args[0];
String fileName = args[1];
String groupName = args[2];
String groupDesc = args[3];
String vpcId = args[4];
String myIpAddress = args[5];

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

SsmClient ssmClient = SsmClient.builder()
    .region(region)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the Amazon EC2 example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);

```

```
        System.out.println("1. Create an RSA key pair and save the private key
material as a .pem file.");
        createKeyPair(ec2, keyName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. List key pairs.");
        describeKeys(ec2);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create a security group.");
        String groupId = createSecurityGroup(ec2, groupName, groupDesc, vpcId,
myIpAddress);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Display security group info for the newly created
security group.");
        describeSecurityGroups(ec2, groupId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Get a list of Amazon Linux 2 AMIs and selects one
with amzn2 in the name.");
        String instanceId = getParaValues(ssmClient);
        System.out.println("The instance Id is " + instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Get more information about an amzn2 image.");
        String amiValue = describeImage(ec2, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Get a list of instance types.");
        String instanceType = getInstanceTypes(ec2);
        System.out.println("The instance type is " + instanceType);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Create an instance.");
        String newInstanceId = runInstance(ec2, instanceType, keyName, groupName,
amiValue);
```

```
System.out.println("The instance Id is " + newInstanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Display information about the running instance. ");
String ipAddress = describeEC2Instances(ec2, newInstanceId);
System.out.println("You can SSH to the instance using this command:");
System.out.println("ssh -i " + fileName + "ec2-user@" + ipAddress);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Stop the instance and use a waiter.");
stopInstance(ec2, newInstanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Start the instance and use a waiter.");
startInstance(ec2, newInstanceId);
ipAddress = describeEC2Instances(ec2, newInstanceId);
System.out.println("You can SSH to the instance using this command:");
System.out.println("ssh -i " + fileName + "ec2-user@" + ipAddress);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Allocate an Elastic IP address and associate it with
the instance.");
String allocationId = allocateAddress(ec2);
System.out.println("The allocation Id value is " + allocationId);
String associationId = associateAddress(ec2, newInstanceId, allocationId);
System.out.println("The associate Id value is " + associationId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("13. Describe the instance again.");
ipAddress = describeEC2Instances(ec2, newInstanceId);
System.out.println("You can SSH to the instance using this command:");
System.out.println("ssh -i " + fileName + "ec2-user@" + ipAddress);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("14. Disassociate and release the Elastic IP address.");
disassociateAddress(ec2, associationId);
releaseEC2Address(ec2, allocationId);
System.out.println(DASHES);
```

```
System.out.println(DASHES);
System.out.println("15. Terminate the instance and use a waiter.");
terminateEC2(ec2, newInstanceId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("16. Delete the security group.");
deleteEC2SecGroup(ec2, groupId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("17. Delete the key.");
deleteKeys(ec2, keyName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("You successfully completed the Amazon EC2 scenario.");
System.out.println(DASHES);
ec2.close();
}

public static void deleteEC2SecGroup(Ec2Client ec2, String groupId) {
    try {
        DeleteSecurityGroupRequest request =
DeleteSecurityGroupRequest.builder()
        .groupId(groupId)
        .build();

        ec2.deleteSecurityGroup(request);
        System.out.println("Successfully deleted security group with Id " +
groupId);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void terminateEC2(Ec2Client ec2, String instanceId) {
    try {
        Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
```

```
        .build());

    TerminateInstancesRequest ti = TerminateInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to
terminate. This will take a few minutes.");
    ec2.terminateInstances(ti);
    DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    WaiterResponse<DescribeInstancesResponse> waiterResponse = ec2Waiter
        .waitUntilInstanceTerminated(instanceRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Successfully started instance " + instanceId);
    System.out.println(instanceId + " is terminated!");

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteKeys(Ec2Client ec2, String keyPair) {
    try {
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
            .keyName(keyPair)
            .build();

        ec2.deleteKeyPair(request);
        System.out.println("Successfully deleted key pair named " + keyPair);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void releaseEC2Address(Ec2Client ec2, String allocId) {
    try {
        ReleaseAddressRequest request = ReleaseAddressRequest.builder()
```



```
        .allocationId(allocId)
        .build();

        ec2.releaseAddress(request);
        System.out.println("Successfully released Elastic IP address " +
allocId);
    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void disassociateAddress(Ec2Client ec2, String associationId) {
    try {
        DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
            .associationId(associationId)
            .build();

        ec2.disassociateAddress(addressRequest);
        System.out.println("You successfully disassociated the address!");

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String associateAddress(Ec2Client ec2, String instanceId, String
allocationId) {
    try {
        AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
            .instanceId(instanceId)
            .allocationId(allocationId)
            .build();

        AssociateAddressResponse associateResponse =
ec2.associateAddress(associateRequest);
        return associateResponse.associationId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }
    return "";
}

public static String allocateAddress(Ec2Client ec2) {
    try {
        AllocateAddressRequest allocateRequest =
AllocateAddressRequest.builder()
            .domain(DomainType.VPC)
            .build();

        AllocateAddressResponse allocateResponse =
ec2.allocateAddress(allocateRequest);
        return allocateResponse.allocationId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void startInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();

    StartInstancesRequest request = StartInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to run. This
will take a few minutes.");
    ec2.startInstances(request);
    DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceRunning(instanceRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Successfully started instance " + instanceId);
}
```

```
}

public static void stopInstance(Ec2Client ec2, String instanceId) {
    Ec2Waiter ec2Waiter = Ec2Waiter.builder()
        .overrideConfiguration(b -> b.maxAttempts(100))
        .client(ec2)
        .build();
    StopInstancesRequest request = StopInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    System.out.println("Use an Ec2Waiter to wait for the instance to stop. This
will take a few minutes.");
    ec2.stopInstances(request);
    DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceStopped(instanceRequest);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Successfully stopped instance " + instanceId);
}

public static String describeEC2Instances(Ec2Client ec2, String newInstanceId) {
    try {
        String pubAddress = "";
        boolean isRunning = false;
        DescribeInstancesRequest request = DescribeInstancesRequest.builder()
            .instanceIds(newInstanceId)
            .build();

        while (!isRunning) {
            DescribeInstancesResponse response = ec2.describeInstances(request);
            String state =
response.reservations().get(0).instances().get(0).state().name().name();
            if (state.compareTo("RUNNING") == 0) {
                System.out.println("Image id is " +
response.reservations().get(0).instances().get(0).imageId());
                System.out.println(
                    "Instance type is " +
response.reservations().get(0).instances().get(0).instanceType());
                System.out.println(
```

```
        "Instance state is " +
response.reservations().get(0).instances().get(0).state().name());
        pubAddress =
response.reservations().get(0).instances().get(0).publicIpAddress();
        System.out.println("Instance address is " + pubAddress);
        isRunning = true;
    }
}
return pubAddress;
} catch (SsmException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return "";
}

public static String runInstance(Ec2Client ec2, String instanceType, String
keyName, String groupName,
    String amiId) {
    try {
        RunInstancesRequest runRequest = RunInstancesRequest.builder()
            .instanceType(instanceType)
            .keyName(keyName)
            .securityGroups(groupName)
            .maxCount(1)
            .minCount(1)
            .imageId(amiId)
            .build();

        System.out.println("Going to start an EC2 instance using a waiter");
        RunInstancesResponse response = ec2.runInstances(runRequest);
        String instanceIdVal = response.instances().get(0).instanceId();
        ec2.waiter().waitUntilInstanceRunning(r ->
r.instanceIds(instanceIdVal));
        System.out.println("Successfully started EC2 instance " + instanceIdVal
+ " based on AMI " + amiId);
        return instanceIdVal;

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

```
// Get a list of instance types.
public static String getInstanceTypes(Ec2Client ec2) {
    String instanceType;
    try {
        DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
        .maxResults(10)
        .build();

        DescribeInstanceTypesResponse response =
ec2.describeInstanceTypes(typesRequest);
        List<InstanceTypeInfo> instanceTypes = response.instanceTypes();
        for (InstanceTypeInfo type : instanceTypes) {
            System.out.println("The memory information of this type is " +
type.memoryInfo().sizeInMiB());
            System.out.println("Network information is " +
type.networkInfo().toString());
            System.out.println("Instance type is " +
type.instanceType().toString());
            instanceType = type.instanceType().toString();
            if (instanceType.compareTo("t2.2xlarge") == 0){
                return instanceType;
            }
        }

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Display the Description field that corresponds to the instance Id value.
public static String describeImage(Ec2Client ec2, String instanceId) {
    try {
        DescribeImagesRequest imagesRequest = DescribeImagesRequest.builder()
        .imageIds(instanceId)
        .build();

        DescribeImagesResponse response = ec2.describeImages(imagesRequest);
        System.out.println("The description of the first image is " +
response.images().get(0).description());
    }
}
```

```
        System.out.println("The name of the first image is " +
response.images().get(0).name());

        // Return the image Id value.
        return response.images().get(0).imageId();

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Get the Id value of an instance with amzn2 in the name.
public static String getParaValues(SsmClient ssmClient) {
    try {
        GetParametersByPathRequest parameterRequest =
GetParametersByPathRequest.builder()
            .path("/aws/service/ami-amazon-linux-latest")
            .build();

        GetParametersByPathIterable responses =
ssmClient.getParametersByPathPaginator(parameterRequest);
        for
(software.amazon.awssdk.services.ssm.model.GetParametersByPathResponse response :
responses) {
            System.out.println("Test " + response.nextToken());
            List<Parameter> parameterList = response.parameters();
            for (Parameter para : parameterList) {
                System.out.println("The name of the para is: " + para.name());
                System.out.println("The type of the para is: " + para.type());
                if (filterName(para.name())) {
                    return para.value();
                }
            }
        }

    } catch (SsmException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

```
// Return true if the name has amzn2 in it. For example:
// /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-arm64-gp2
private static boolean filterName(String name) {
    String[] parts = name.split("/");
    String myValue = parts[4];
    return myValue.contains("amzn2");
}

public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
    try {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
            .groupIds(groupId)
            .build();

        // Use a paginator.
        DescribeSecurityGroupsIterable listGroups =
ec2.describeSecurityGroupsPaginator(request);
        listGroups.stream()
            .flatMap(r -> r.securityGroups().stream())
            .forEach(group -> System.out
                .println(" Group id: " +group.groupId() + " group name = " +
group.groupName()));

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String createSecurityGroup(Ec2Client ec2, String groupName, String
groupDesc, String vpcId,
    String myIpAddress) {
    try {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

        CreateSecurityGroupResponse resp =
ec2.createSecurityGroup(createRequest);
        IpRange ipRange = IpRange.builder()
```

```
        .cidrIp(myIpAddress + "/0")
        .build();

    IpPermission ipPerm = IpPermission.builder()
        .ipProtocol("tcp")
        .toPort(80)
        .fromPort(80)
        .ipRanges(ipRange)
        .build();

    IpPermission ipPerm2 = IpPermission.builder()
        .ipProtocol("tcp")
        .toPort(22)
        .fromPort(22)
        .ipRanges(ipRange)
        .build();

    AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
        .groupName(groupName)
        .ipPermissions(ipPerm, ipPerm2)
        .build();

    ec2.authorizeSecurityGroupIngress(authRequest);
    System.out.println("Successfully added ingress policy to security group
" + groupName);
    return resp.groupId();

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void describeKeys(Ec2Client ec2) {
    try {
        DescribeKeyPairsResponse response = ec2.describeKeyPairs();
        response.keyPairs().forEach(keyPair -> System.out.printf(
            "Found key pair with name %s " +
                "and fingerprint %s",
            keyPair.keyName(),
            keyPair.keyFingerprint()));
    }
```



```
        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createKeyPair(Ec2Client ec2, String keyName, String fileName)
    {
        try {
            CreateKeyPairRequest request = CreateKeyPairRequest.builder()
                .keyName(keyName)
                .build();

            CreateKeyPairResponse response = ec2.createKeyPair(request);
            String content = response.keyMaterial();
            BufferedWriter writer = new BufferedWriter(new FileWriter(fileName));
            writer.write(content);
            writer.close();
            System.out.println("Successfully created key pair named " + keyName);

        } catch (Ec2Exception | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

• 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。

- [AllocateAddress](#)
- [AssociateAddress](#)
- [AuthorizeSecurityGroupIngress](#)
- [CreateKeyPair](#)
- [CreateSecurityGroup](#)
- [DeleteKeyPair](#)
- [DeleteSecurityGroup](#)
- [DescribeImages](#)
- [DescribeInstanceTypes](#)
- [DescribeInstances](#)

- [DescribeKeyPairs](#)
- [DescribeSecurityGroups](#)
- [DisassociateAddress](#)
- [ReleaseAddress](#)
- [RunInstances](#)
- [StartInstances](#)
- [StopInstances](#)
- [TerminateInstances](#)
- [UnmonitorInstances](#)

使用SDK适用于 Java 2.x 的亚马逊ECR示例

以下代码示例向您展示如何在 Amazon 中使用来执行操作和实现常见场景 ECR。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

开始使用

你好 Amazon ECR

以下代码示例展示了如何开始使用 Amazon ECR。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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()));
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [listImages](#)”中的。

主题

- [操作](#)
- [场景](#)

操作

CreateRepository

以下代码示例演示如何使用 CreateRepository。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

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

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateRepository](#)”中的。

DeleteRepository

以下代码示例演示如何使用 DeleteRepository。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**
 * 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();
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteRepository](#)”中的。

DescribeImages

以下代码示例演示如何使用 DescribeImages。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```

/**
 * 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();
    }

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeImages](#)”中的。

DescribeRepositories

以下代码示例演示如何使用 DescribeRepositories。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**
```



```

    * 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();
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeRepositories](#)”中的。

GetAuthorizationToken

以下代码示例演示如何使用 GetAuthorizationToken。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```

/**
 * 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();
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetAuthorizationToken](#)”中的。

GetRepositoryPolicy

以下代码示例演示如何使用 GetRepositoryPolicy。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```

/**
 * 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;
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetRepositoryPolicy](#)”中的。

PushImageCmd

以下代码示例演示如何使用 PushImageCmd。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```

/**
 * 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();
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [PushImageCmd](#)”中的。

SetRepositoryPolicy

以下代码示例演示如何使用 SetRepositoryPolicy。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```

/**
 * 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();
    }
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [SetRepositoryPolicy](#)”中的。

StartLifecyclePolicyPreview

以下代码示例演示如何使用 StartLifecyclePolicyPreview。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**
 * 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();
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartLifecyclePolicyPreview](#)”中的。

场景

了解亚马逊的ECR核心运营

以下代码示例展示了如何：

- 创建亚马逊ECR存储库。
- 设置存储库策略。
- 检索存储库URIs。
- 获取亚马逊ECR授权令牌。
- 为 Amazon ECR 存储库设置生命周期策略。
- 将 Docker 镜像推送到亚马逊ECR存储库。
- 验证 Amazon ECR 存储库中是否存在图片。
- 列出您账户的 Amazon ECR 存储库并获取有关这些存储库的详细信息。
- 删除亚马逊ECR存储库。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

运行演示 Amazon ECR 功能的交互式场景。

```
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.RepositoryPolicyNotFoundException;

import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code example requires an IAM Role that has permissions to interact with
 * the Amazon ECR service.
 *
 * To create an IAM role, see:
 *
 * https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create.html
 *
 * This Java scenario example requires a local docker image named echo-text. Without
 * a local image,
 * this Java program will not successfully run. For more information including how
 * to create the local
 * image, see:
 *
 * /getting_started_scenarios/ecr_scenario/README
 */
public class ECRScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) {
        final String usage = ""
```

```
Usage: <iamRoleARN> <accountId>

Where:
    iamRoleARN - The IAM role ARN that has the necessary permissions to
access and manage the Amazon ECR repository.
    accountId - Your AWS account number.
""";

if (args.length != 2) {
    System.out.println(usage);
    return;
}

ECRActions ecrActions = new ECRActions();
String iamRole = args[0];
String accountId = args[1];
String localImageName;

Scanner scanner = new Scanner(System.in);
System.out.println("""
    The Amazon Elastic Container Registry (ECR) is a fully-managed Docker
container registry
    service provided by AWS. It allows developers and organizations to
securely
    store, manage, and deploy Docker container images.
    ECR provides a simple and scalable way to manage container images
throughout their lifecycle,
    from building and testing to production deployment.\s

    The `EcrAsyncClient` interface in the AWS SDK for Java 2.x provides a
set of methods to
    programmatically interact with the Amazon ECR service. This allows
developers to
    automate the storage, retrieval, and management of container images as
part of their application
    deployment pipelines. With ECR, teams can focus on building and
deploying their
    applications without having to worry about the underlying
infrastructure required to
    host and manage a container registry.

    This scenario walks you through how to perform key operations for this
service.

    Let's get started...
```

```

        You have two choices:
        1 - Run the entire program.
        2 - Delete an existing Amazon ECR repository named echo-text (created
from a previous execution of
        this program that did not complete).
        """);

while (true) {
    String input = scanner.nextLine();
    if (input.trim().equalsIgnoreCase("1")) {
        System.out.println("Continuing with the program...");
        System.out.println("");
        break;
    } else if (input.trim().equalsIgnoreCase("2")) {
        String repoName = "echo-text";
        ecrActions.deleteECRRepository(repoName);
        return;
    } else {
        // Handle invalid input.
        System.out.println("Invalid input. Please try again.");
    }
}

waitForInputToContinue(scanner);
System.out.println(DASHES);

System.out.println("""
    1. Create an ECR repository.

    The first task is to ensure we have a local Docker image named echo-
text.

    If this image exists, then an Amazon ECR repository is created.

    An ECR repository is a private Docker container repository provided
    by Amazon Web Services (AWS). It is a managed service that makes it easy
    to store, manage, and deploy Docker container images.\s
    """);

// Ensure that a local docker image named echo-text exists.
boolean doesExist = ecrActions.isEchoTextImagePresent();
String repoName;
if (!doesExist){
    System.out.println("The local image named echo-text does not exist");
}

```

```

        return;
    } else {
        localImageName = "echo-text";
        repoName = "echo-text";
    }

    try {
        String repoArn = ecrActions.createECRRepository(repoName);
        System.out.println("The ARN of the ECR repository is " + repoArn);

    } catch (IllegalArgumentException e) {
        System.err.println("Invalid repository name: " + e.getMessage());
        return;
    } catch (RuntimeException e) {
        System.err.println("An error occurred while creating the ECR repository:
" + e.getMessage());
        e.printStackTrace();
        return;
    }
    waitForInputToContinue(scanner);

    System.out.println(DASHES);
    System.out.println("""
2. Set an ECR repository policy.

Setting an ECR repository policy using the `setRepositoryPolicy` function is
crucial for maintaining
the security and integrity of your container images. The repository policy
allows you to
define specific rules and restrictions for accessing and managing the images
stored within your ECR
repository.
""");
    waitForInputToContinue(scanner);
    try {
        ecrActions.setRepoPolicy(repoName, iamRole);

    } catch (RepositoryPolicyNotFoundException e) {
        System.err.println("Invalid repository name: " + e.getMessage());
        return;
    } catch (EcrException e) {
        System.err.println("An ECR exception occurred: " + e.getMessage());
        return;
    } catch (RuntimeException e) {

```

```
        System.err.println("An error occurred while creating the ECR repository:
" + e.getMessage());
        return;
    }
    waitForInputToContinue(scanner);

    System.out.println(DASHES);
    System.out.println("
3. Display ECR repository policy.

Now we will retrieve the ECR policy to ensure it was successfully set.
");
    waitForInputToContinue(scanner);
    try {
        String policyText = ecrActions.getRepoPolicy(repoName);
        System.out.println("Policy Text:");
        System.out.println(policyText);

    } catch (EcrException e) {
        System.err.println("An ECR exception occurred: " + e.getMessage());
        return;
    } catch (RuntimeException e) {
        System.err.println("An error occurred while creating the ECR repository:
" + e.getMessage());
        return;
    }

    waitForInputToContinue(scanner);

    System.out.println(DASHES);
    System.out.println("
4. Retrieve an ECR authorization token.
```

You need an authorization token to securely access and interact with the Amazon ECR registry.

The `getAuthorizationToken` method of the `EcrAsyncClient` is responsible for securely accessing and interacting with an Amazon ECR repository. This operation is responsible for obtaining a valid authorization token, which is required to authenticate your requests to the ECR service.

Without a valid authorization token, you would not be able to perform any operations on the

```

    ECR repository, such as pushing, pulling, or managing your Docker images.

    """);
    waitForInputToContinue(scanner);
    try {
        ecrActions.getAuthToken();

    } catch (EcrException e) {
        System.err.println("An ECR exception occurred: " + e.getMessage());
        return;
    } catch (RuntimeException e) {
        System.err.println("An error occurred while retrieving the authorization
token: " + e.getMessage());
        return;
    }
    waitForInputToContinue(scanner);

    System.out.println(DASHES);
    System.out.println("""
5. Get the ECR Repository URI.

    The URI of an Amazon ECR repository is important. When you want to deploy a
container image to
    a container orchestration platform like Amazon Elastic Kubernetes Service
(EKS)
    or Amazon Elastic Container Service (ECS), you need to specify the full
image URI,
    which includes the ECR repository URI. This allows the container runtime to
pull the
    correct container image from the ECR repository.
    """);
    waitForInputToContinue(scanner);

    try {
        ecrActions.getRepositoryURI(repoName);

    } catch (EcrException e) {
        System.err.println("An ECR exception occurred: " + e.getMessage());
        return;
    }

    } catch (RuntimeException e) {
        System.err.println("An error occurred while retrieving the URI: " +
e.getMessage());
        return;
    }

```

```

    }
    waitForInputToContinue(scanner);

    System.out.println(DASHES);
    System.out.println("""
        6. Set an ECR Lifecycle Policy.

        An ECR Lifecycle Policy is used to manage the lifecycle of Docker images
        stored in your ECR repositories.
        These policies allow you to automatically remove old or unused Docker
        images from your repositories,
        freeing up storage space and reducing costs.

        This example policy helps to maintain the size and efficiency of the
        container registry
        by automatically removing older and potentially unused images, ensuring
        that the
        storage is optimized and the registry remains up-to-date.
        """);
    waitForInputToContinue(scanner);
    try {
        ecrActions.setLifecyclePolicy(repoName);

    } catch (RuntimeException e) {
        System.err.println("An error occurred while setting the lifecycle
        policy: " + e.getMessage());
        e.printStackTrace();
        return;
    }
    waitForInputToContinue(scanner);

    System.out.println(DASHES);
    System.out.println("""
        7. Push a docker image to the Amazon ECR Repository.

        The `pushImageCmd()` method pushes a local Docker image to an Amazon ECR
        repository.
        It sets up the Docker client by connecting to the local Docker host using
        the default port.
        It then retrieves the authorization token for the ECR repository by making a
        call to the AWS SDK.

        The method uses the authorization token to create an `AuthConfig` object,
        which is used to authenticate

```


the Docker client when pushing the image. Finally, the method tags the Docker image with the specified repository name and image tag, and then pushes the image to the ECR repository using the Docker client.

If the push operation is successful, the method prints a message indicating that the image was pushed to ECR.

```
        """);
        waitForInputToContinue(scanner);

        try {
            ecrActions.pushDockerImage(repoName, localImageName);

        } catch (RuntimeException e) {
            System.err.println("An error occurred while pushing a local Docker image
to Amazon ECR: " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("8. Verify if the image is in the ECR Repository.");
        waitForInputToContinue(scanner);
        try {
            ecrActions.verifyImage(repoName, localImageName);

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("9. As an optional step, you can interact with the image
in Amazon ECR by using the CLI.");
        System.out.println("Would you like to view instructions on how to use the
CLI to run the image? (y/n)");
        String ans = scanner.nextLine().trim();
        if (ans.equalsIgnoreCase("y")) {
            String instructions = ""
```

1. Authenticate with ECR - Before you can pull the image from Amazon ECR, you need to authenticate with the registry. You can do this using the AWS CLI:

```
aws ecr get-login-password --region us-east-1 | docker login --
username AWS --password-stdin %s.dkr.ecr.us-east-1.amazonaws.com
```

2. Describe the image using this command:

```
aws ecr describe-images --repository-name %s --image-ids imageTag=%s
```

3. Run the Docker container and view the output using this command:

```
docker run --rm %s.dkr.ecr.us-east-1.amazonaws.com/%s:%s
""";
```

```
instructions = String.format(instructions, accountId, repoName,
localImageName, accountId, repoName, localImageName);
```

```
System.out.println(instructions);
```

```
}
```

```
waitForInputToContinue(scanner);
```

```
System.out.println(DASHES);
```

```
System.out.println("10. Delete the ECR Repository.");
```

```
System.out.println(
```

```
""";
```

If the repository isn't empty, you must either delete the contents of the repository

or use the force option (used in this scenario) to delete the repository and have Amazon ECR delete all of its contents

on your behalf.

```
""");
```

```
System.out.println("Would you like to delete the Amazon ECR Repository? (y/
n)");
```

```
String delAns = scanner.nextLine().trim();
```

```
if (delAns.equalsIgnoreCase("y")) {
```

```
System.out.println("You selected to delete the AWS ECR resources.");
```

```
try {
```

```
ecrActions.deleteECRRepository(repoName);
```

```
} catch (EcrException e) {
```

```
System.err.println("An ECR exception occurred: " + e.getMessage());
```

```
return;
```

```
} catch (RuntimeException e) {
```

```

        System.err.println("An error occurred while deleting the Docker
image: " + e.getMessage());
        e.printStackTrace();
        return;
    }
}

System.out.println(DASHES);
System.out.println("This concludes the Amazon ECR SDK scenario");
System.out.println(DASHES);
}

private static void waitForInputToContinue(Scanner scanner) {
    while (true) {
        System.out.println("");
        System.out.println("Enter 'c' followed by <ENTER> to continue:");
        String input = scanner.nextLine();

        if (input.trim().equalsIgnoreCase("c")) {
            System.out.println("Continuing with the program...");
            System.out.println("");
            break;
        } else {
            // Handle invalid input.
            System.out.println("Invalid input. Please try again.");
        }
    }
}
}
}

```

Amazon ECR SDK 方法的包装器类。

```

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;
    }
}
```



```
    }  
  }  
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [CreateRepository](#)
 - [DeleteRepository](#)
 - [DescribeImages](#)
 - [DescribeRepositories](#)
 - [GetAuthorizationToken](#)
 - [GetRepositoryPolicy](#)
 - [SetRepositoryPolicy](#)
 - [StartLifecyclePolicyPreview](#)

使用SDK适用于 Java 2.x 的亚马逊ECS示例

以下代码示例向您展示如何在 Amazon 中使用来执行操作和实现常见场景ECS。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

CreateCluster

以下代码示例演示如何使用 CreateCluster。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateCluster](#)”中的。

CreateService

以下代码示例演示如何使用 CreateService。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateService](#)”中的。

DeleteService

以下代码示例演示如何使用 DeleteService。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteService](#)”中的。

DescribeClusters

以下代码示例演示如何使用 DescribeClusters。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

```
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeClusters](#)”中的。

DescribeTasks

以下代码示例演示如何使用 DescribeTasks。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeTasks](#)”中的。

ListClusters

以下代码示例演示如何使用 ListClusters。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListClusters](#)”中的。

UpdateService

以下代码示例演示如何使用 UpdateService。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```

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);
    }
}
```

```
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [UpdateService](#)”中的。

Elastic Load Balancing—— SDK 适用于 Java 2.x 的版本 2 示例

以下代码示例向您展示了如何在 Elastic Load Balancing-版本 2 中 AWS SDK for Java 2.x 使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

开始使用

开始使用 Elastic Load Balancing

以下代码示例演示了如何开始使用 Elastic Load Balancing。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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());
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeLoadBalancers](#)”中的。

主题

- [操作](#)
- [场景](#)

操作

CreateListener

以下代码示例演示如何使用 CreateListener。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/*
 * Creates an Elastic Load Balancing load balancer that uses the specified
 * subnets
 * and forwards requests to the specified target group.
 */
public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
    String protocol) {
    try {
        List<String> subnetIdStrings = subnetIds.stream()
```



```
        .map(Subnet::subnetId)
        .collect(Collectors.toList());

    CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
        .subnets(subnetIdStrings)
        .name(lbName)
        .scheme("internet-facing")
        .build();

    // Create and wait for the load balancer to become available.
    CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
    String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

    ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
    DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
        .loadBalancerArns(lbARN)
        .build();

    System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
    WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
        .waitUntilLoadBalancerAvailable(request);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Load Balancer " + lbName + " is available.");

    // Get the DNS name (endpoint) of the load balancer.
    String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
    System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

    // Create a listener for the load balance.
    Action action = Action.builder()
        .targetGroupArn(targetGroupARN)
        .type("forward")
        .build();

    CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
        .defaultActions(action)
```

```
        .port(port)
        .protocol(protocol)
        .defaultActions(action)
        .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
        + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateListener](#)”中的。

CreateLoadBalancer

以下代码示例演示如何使用 CreateLoadBalancer。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/*
 * Creates an Elastic Load Balancing load balancer that uses the specified
 * subnets
 * and forwards requests to the specified target group.
 */
public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
String protocol) {
```

```
try {
    List<String> subnetIdStrings = subnetIds.stream()
        .map(Subnet::subnetId)
        .collect(Collectors.toList());

    CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
    .subnets(subnetIdStrings)
    .name(lbName)
    .scheme("internet-facing")
    .build();

    // Create and wait for the load balancer to become available.
    CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
    String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

    ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
    DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
    .loadBalancerArns(lbARN)
    .build();

    System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
    WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
        .waitUntilLoadBalancerAvailable(request);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Load Balancer " + lbName + " is available.");

    // Get the DNS name (endpoint) of the load balancer.
    String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
    System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

    // Create a listener for the load balance.
    Action action = Action.builder()
        .targetGroupArn(targetGroupARN)
        .type("forward")
        .build();

    CreateListenerRequest listenerRequest = CreateListenerRequest.builder()
```

```
.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
    .defaultActions(action)
    .port(port)
    .protocol(protocol)
    .defaultActions(action)
    .build();

    getLoadBalancerClient().createListener(listenerRequest);
    System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
    + targetGroupARN);

    // Return the load balancer DNS name.
    return lbDNSName;

} catch (ElasticLoadBalancingV2Exception e) {
    e.printStackTrace();
}
return "";
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateLoadBalancer](#)”中的。

CreateTargetGroup

以下代码示例演示如何使用 CreateTargetGroup。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/*
 * 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;
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateTargetGroup](#)”中的。

DeleteLoadBalancer

以下代码示例演示如何使用 DeleteLoadBalancer。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
// 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.");
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteLoadBalancer](#)”中的。

DeleteTargetGroup

以下代码示例演示如何使用 DeleteTargetGroup。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```

// 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.");
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteTargetGroup](#)”中的。

DescribeTargetHealth

以下代码示例演示如何使用 DescribeTargetHealth。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

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

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeTargetHealth](#)”中的。

场景

构建和管理弹性服务

以下代码示例演示了如何创建可返回书籍、电影和歌曲推荐的负载均衡的 Web 服务。该示例演示服务如何响应故障，以及如何重组服务以提高故障发生时的弹性。

- 使用 Amazon A EC2 uto Scaling 组根据启动模板创建亚马逊弹性计算云 (AmazonEC2) 实例，并将实例数量保持在指定范围内。
- 使用 Elastic Load Balancing 处理和分发HTTP请求。
- 监控自动扩缩组中实例的运行状况，并仅将请求转发到运行状况良好的实例。
- 在每个EC2实例上运行 Python 网络服务器来处理HTTP请求。Web 服务器以建议和运行状况检查作为响应。
- 使用 Amazon DynamoDB 表模拟推荐服务。
- 通过更新 AWS Systems Manager 参数来控制 Web 服务器对请求和运行状况检查的响应。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

在命令提示符中运行交互式场景。

```
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.deleteAutoScalingGroup(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 wasSuccessful = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
if (!wasSuccessful) {
    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 (wasSuccessful) {
    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);  
  }  
}
```

创建一个包含 Auto Scaling 和 Amazon EC2 操作的类。

```
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.getInstanceInformationList();
        for (InstanceInformation info : instanceInformationList) {
            if (info.getInstanceId().equals(instanceId)) {
                instReady = true;
                break;
            }
        }
    }

    SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
        .instanceIds(instanceId)
        .documentName("AWS-RunShellScript")
        .parameters(Collections.singletonMap("commands",
80")))
            Collections.singletonList("cd / && sudo python3 server.py

        .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.");
}
}

```

创建一个包含弹性负载均衡操作的类。

```
public class LoadBalancer {
```

```
public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

public ElasticLoadBalancingV2Client getLoadBalancerClient() {
    if (elasticLoadBalancingV2Client == null) {
        elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }

    return elasticLoadBalancingV2Client;
}

// Checks the health of the instances in the target group.
public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
    DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
        .names(targetGroupName)
        .build();

    DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

    DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
        .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
        .build();

    DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
    return healthResponse.targetHealthDescriptions();
}

// Gets the HTTP endpoint of the load balancer.
public String getEndpoint(String lbName) {
    DescribeLoadBalancersResponse res = getLoadBalancerClient()
        .describeLoadBalancers(describe -> describe.names(lbName));
    return res.loadBalancers().get(0).dnsName();
}

// Deletes a load balancer.
public void deleteLoadBalancer(String lbName) {
    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
```

```

        .describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
            .build();

        getLoadBalancerClient().deleteLoadBalancer(
            builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancersDeleted(request);
        waiterResponse.matched().response().ifPresent(System.out::println);

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}

// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
            .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
            .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}

// Verify this computer can successfully send a GET request to the load balancer
// endpoint.
public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
    boolean success = false;
    int retries = 3;
    CloseableHttpClient httpClient = HttpClients.createDefault();

```

```
// Create an HTTP GET request to the ELB.
HttpGet httpGet = new HttpGet("http://" + elbDnsName);
try {
    while ((!success) && (retries > 0)) {
        // Execute the request and get the response.
        HttpResponse response = httpClient.execute(httpGet);
        int statusCode = response.getStatusLine().getStatusCode();
        System.out.println("HTTP Status Code: " + statusCode);
        if (statusCode == 200) {
            success = true;
        } else {
            retries--;
            System.out.println("Got connection error from load balancer
endpoint, retrying...");
            TimeUnit.SECONDS.sleep(15);
        }
    }

    } catch (org.apache.http.conn.HttpHostConnectException e) {
        System.out.println(e.getMessage());
    }

    System.out.println("Status.." + success);
    return success;
}

/*
 * Creates an Elastic Load Balancing target group. The target group specifies
 * how
 * the load balancer forward requests to instances in the group and how instance
 * health is checked.
 */
public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
        .name(targetGroupName)
        .protocol(protocol)
        .build();
}
```

```
        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /**
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                .map(Subnet::subnetId)
                .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                .subnets(subnetIdStrings)
                .name(lbName)
                .scheme("internet-facing")
                .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                .loadBalancerArns(lbARN)
                .build();
```

```
        System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

        // Create a listener for the load balance.
        Action action = Action.builder()
            .targetGroupArn(targetGroupARN)
            .type("forward")
            .build();

        CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
            .defaultActions(action)
            .port(port)
            .protocol(protocol)
            .defaultActions(action)
            .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
            + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
}
```

创建一个使用 DynamoDB 模拟推荐服务的类。

```
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);
}
}
```

创建一个包含 Systems Manager 操作的类。

```
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);
    }
}
```

• 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。

- [AttachLoadBalancerTargetGroups](#)
- [CreateAutoScalingGroup](#)
- [CreateInstanceProfile](#)
- [CreateLaunchTemplate](#)
- [CreateListener](#)

- [CreateLoadBalancer](#)
- [CreateTargetGroup](#)
- [DeleteAutoScalingGroup](#)
- [DeleteInstanceProfile](#)
- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeIamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplaceIamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

MediaStore 用SDK于 Java 2.x 的示例

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 MediaStore。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

CreateContainer

以下代码示例演示如何使用 CreateContainer。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

```
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateContainer](#)”中的。

DeleteContainer

以下代码示例演示如何使用 DeleteContainer。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteContainer](#)”中的。

DeleteObject

以下代码示例演示如何使用 DeleteObject。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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();
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteObject](#)”中的。

DescribeContainer

以下代码示例演示如何使用 DescribeContainer。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeContainer](#)”中的。

GetObject

以下代码示例演示如何使用 GetObject。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 outputStream = new FileOutputStream(targetFile);
        outputStream.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();
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetObject](#)”中的。

ListContainers

以下代码示例演示如何使用 ListContainers。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListContainers](#)”中的。

PutObject

以下代码示例演示如何使用 PutObject。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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();
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [PutObject](#)”中的。

OpenSearch SDK适用于 Java 2.x 的服务示例

以下代码示例向您展示了如何使用 with S OpenSearch ervice 来执行操作和实现常见场景。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

CreateDomain

以下代码示例演示如何使用 CreateDomain。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.ClusterConfig;
import software.amazon.awssdk.services.opensearch.model.EBSOptions;
import software.amazon.awssdk.services.opensearch.model.VolumeType;
import software.amazon.awssdk.services.opensearch.model.NodeToNodeEncryptionOptions;
import software.amazon.awssdk.services.opensearch.model.CreateDomainRequest;
import software.amazon.awssdk.services.opensearch.model.CreateDomainResponse;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateDomain {
```

```
public static void main(String[] args) {
    final String usage = ""

        Usage:
            <domainName>

        Where:
            domainName - The name of the domain to create.
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String domainName = args[0];
    Region region = Region.US_EAST_1;
    OpenSearchClient searchClient = OpenSearchClient.builder()
        .region(region)
        .build();

    createNewDomain(searchClient, domainName);
    System.out.println("Done");
}

public static void createNewDomain(OpenSearchClient searchClient, String
domainName) {
    try {
        ClusterConfig clusterConfig = ClusterConfig.builder()
            .dedicatedMasterEnabled(true)
            .dedicatedMasterCount(3)
            .dedicatedMasterType("t2.small.search")
            .instanceType("t2.small.search")
            .instanceCount(5)
            .build();

        EBSOptions ebsOptions = EBSOptions.builder()
            .ebsEnabled(true)
            .volumeSize(10)
            .volumeType(VolumeType.GP2)
            .build();

        NodeToNodeEncryptionOptions encryptionOptions =
NodeToNodeEncryptionOptions.builder()
```

```
        .enabled(true)
        .build();

        CreateDomainRequest domainRequest = CreateDomainRequest.builder()
            .domainName(domainName)
            .engineVersion("OpenSearch_1.0")
            .clusterConfig(clusterConfig)
            .ebsOptions(ebsOptions)
            .nodeToNodeEncryptionOptions(encryptionOptions)
            .build();

        System.out.println("Sending domain creation request...");
        CreateDomainResponse createResponse =
searchClient.createDomain(domainRequest);
        System.out.println("Domain status is " +
createResponse.domainStatus().toString());
        System.out.println("Domain Id is " +
createResponse.domainStatus().domainId());

    } catch (OpenSearchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateDomain](#)”中的。

DeleteDomain

以下代码示例演示如何使用 DeleteDomain。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;
import software.amazon.awssdk.services.opensearch.model.DeleteDomainRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteDomain {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <domainName>

            Where:
                domainName - The name of the domain to delete.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainName = args[0];
        Region region = Region.US_EAST_1;
        OpenSearchClient searchClient = OpenSearchClient.builder()
            .region(region)
            .build();

        deleteSpecificDomain(searchClient, domainName);
        System.out.println("Done");
    }

    public static void deleteSpecificDomain(OpenSearchClient searchClient, String
domainName) {
        try {
            DeleteDomainRequest domainRequest = DeleteDomainRequest.builder()
                .domainName(domainName)
                .build();
```

```
        searchClient.deleteDomain(domainRequest);
        System.out.println(domainName + " was successfully deleted.");

    } catch (OpenSearchException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteDomain](#)”中的。

ListDomainNames

以下代码示例演示如何使用 ListDomainNames。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.DomainInfo;
import software.amazon.awssdk.services.opensearch.model.ListDomainNamesRequest;
import software.amazon.awssdk.services.opensearch.model.ListDomainNamesResponse;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class ListDomainNames {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        OpenSearchClient searchClient = OpenSearchClient.builder()
            .region(region)
            .credentialsProvider(ProfileCredentialsProvider.create())
            .build();
        listAllDomains(searchClient);
        System.out.println("Done");
    }

    public static void listAllDomains(OpenSearchClient searchClient) {
        try {
            ListDomainNamesRequest namesRequest = ListDomainNamesRequest.builder()
                .engineType("OpenSearch")
                .build();

            ListDomainNamesResponse response =
searchClient.listDomainNames(namesRequest);
            List<DomainInfo> domainInfoList = response.domainNames();
            for (DomainInfo domain : domainInfoList)
                System.out.println("Domain name is " + domain.domainName());


        } catch (OpenSearchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListDomainNames](#)”中的。

UpdateDomainConfig

以下代码示例演示如何使用 UpdateDomainConfig。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchClient;
import software.amazon.awssdk.services.opensearch.model.ClusterConfig;
import software.amazon.awssdk.services.opensearch.model.OpenSearchException;
import software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigRequest;
import software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class UpdateDomain {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <domainName>

                Where:
                domainName - The name of the domain to update.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainName = args[0];
        Region region = Region.US_EAST_1;
```

```
        OpenSearchClient searchClient = OpenSearchClient.builder()
            .region(region)
            .build();

        updateSpecificDomain(searchClient, domainName);
        System.out.println("Done");
    }

    public static void updateSpecificDomain(OpenSearchClient searchClient, String
domainName) {
        try {
            ClusterConfig clusterConfig = ClusterConfig.builder()
                .instanceCount(3)
                .build();

            UpdateDomainConfigRequest updateDomainConfigRequest =
UpdateDomainConfigRequest.builder()
                .domainName(domainName)
                .clusterConfig(clusterConfig)
                .build();

            System.out.println("Sending domain update request...");
            UpdateDomainConfigResponse updateResponse =
searchClient.updateDomainConfig(updateDomainConfigRequest);
            System.out.println("Domain update response from Amazon OpenSearch
Service:");
            System.out.println(updateResponse.toString());

        } catch (OpenSearchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [UpdateDomainConfig](#)”中的。

EventBridge 用SDK于 Java 2.x 的示例

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 EventBridge。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。


每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

开始使用

你好 EventBridge

以下代码示例展示了如何开始使用 EventBridge。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * 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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListEventBuses](#)”中的。

主题

- [操作](#)
- [场景](#)

操作

DeleteRule

以下代码示例演示如何使用 DeleteRule。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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");
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteRule](#)”中的。

DescribeRule

以下代码示例演示如何使用 DescribeRule。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeRule](#)”中的。

DisableRule

以下代码示例演示如何使用 `DisableRule`。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用规则名称禁用规则。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DisableRule](#)”中的。

EnableRule

以下代码示例演示如何使用 EnableRule。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用规则名称启用规则。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [EnableRule](#)”中的。

ListRuleNamesByTarget

以下代码示例演示如何使用 ListRuleNamesByTarget。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

列出使用此目标的所有规则名称。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListRuleNamesByTarget](#)”中的。

ListRules

以下代码示例演示如何使用 ListRules。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用规则名称启用规则。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListRules](#)”中的。

ListTargetsByRule

以下代码示例演示如何使用 ListTargetsByRule。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用规则名称列出规则的所有目标。

```
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());
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListTargetsByRule](#)”中的。

PutEvents

以下代码示例演示如何使用 PutEvents。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [PutEvents](#)”中的。

PutRule

以下代码示例演示如何使用 PutRule。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

创建计划规则。

```
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);
    }
}

```

创建规则，将对象添加到 Amazon Simple Storage Service 桶时触发。

```

// 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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [PutRule](#)”中的。

PutTargets

以下代码示例演示如何使用 PutTargets。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

将 Amazon SNS 主题添加为规则的目标。

```
// 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 + ".");
}
```

将输入转换器添加到规则的目标。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [PutTargets](#)”中的。

RemoveTargets

以下代码示例演示如何使用 RemoveTargets。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用规则名称删除该规则的所有目标。

```
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");
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [RemoveTargets](#)”中的。

场景

开始使用规则和目标

以下代码示例演示了操作流程：

- 创建规则并为其添加目标。
- 启用和禁用规则。
- 列出并更新规则和目标。
- 发送事件，然后清理资源。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including 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.getAbsolutePath());
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 "";
}
}
```


• 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。

- [DeleteRule](#)
- [DescribeRule](#)
- [DisableRule](#)
- [EnableRule](#)
- [ListRuleNamesByTarget](#)
- [ListRules](#)
- [ListTargetsByRule](#)
- [PutEvents](#)
- [PutRule](#)
- [PutTargets](#)

将事件通知发送至 EventBridge

以下代码示例演示如何使用存储桶能够将 S3 事件通知发送到亚马逊主题 EventBridge 和亚马逊队列，以及如何将通知发送到亚马逊SNS主题和亚马逊SQS队列。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/** 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;
}

```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [PutBucketNotificationConfiguration](#)
 - [PutRule](#)
 - [PutTargets](#)

使用SDK适用于 Java 2.x 的 Forecast 示例

以下代码示例向您展示了如何使用 with Forecast 来执行操作和实现常见场景。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

CreateDataset

以下代码示例演示如何使用 CreateDataset。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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;
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateDataset](#)”中的。

CreateForecast

以下代码示例演示如何使用 CreateForecast。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateForecast](#)”中的。

DeleteDataset

以下代码示例演示如何使用 DeleteDataset。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteDataset](#)”中的。

DeleteForecast

以下代码示例演示如何使用 DeleteForecast。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteForecast](#)”中的。

DescribeForecast

以下代码示例演示如何使用 DescribeForecast。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeForecast](#)”中的。

ListDatasetGroups

以下代码示例演示如何使用 ListDatasetGroups。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListDatasetGroups](#)”中的。

ListForecasts

以下代码示例演示如何使用 ListForecasts。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```

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);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListForecasts](#)”中的。

AWS Glue 用SDK于 Java 2.x 的示例

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS Glue。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

开始使用

你好 AWS Glue

以下代码示例展示了如何开始使用 AWS Glue。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

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

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListJobs](#)”中的。

主题

- [操作](#)
- [场景](#)

操作

CreateCrawler

以下代码示例演示如何使用 CreateCrawler。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.CreateCrawlerRequest;
import software.amazon.awssdk.services.glue.model.CrawlerTargets;
import software.amazon.awssdk.services.glue.model.GlueException;
import software.amazon.awssdk.services.glue.model.S3Target;
```

```
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateCrawler {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <IAM> <s3Path> <cron> <dbName> <crawlerName>

            Where:
                IAM - The ARN of the IAM role that has AWS Glue and S3
permissions.\s
                s3Path - The Amazon Simple Storage Service (Amazon S3) target
that contains data (for example, CSV data).
                cron - A cron expression used to specify the schedule (i.e.,
cron(15 12 * * ? *).
                dbName - The database name.\s
                crawlerName - The name of the crawler.\s
            """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String iam = args[0];
        String s3Path = args[1];
        String cron = args[2];
        String dbName = args[3];
        String crawlerName = args[4];
        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
            .region(region)
            .build();
```

```
        createGlueCrawler(glueClient, iam, s3Path, cron, dbName, crawlerName);
        glueClient.close();
    }

    public static void createGlueCrawler(GlueClient glueClient,
        String iam,
        String s3Path,
        String cron,
        String dbName,
        String crawlerName) {

        try {
            S3Target s3Target = S3Target.builder()
                .path(s3Path)
                .build();

            // Add the S3Target to a list.
            List<S3Target> targetList = new ArrayList<>();
            targetList.add(s3Target);

            CrawlerTargets targets = CrawlerTargets.builder()
                .s3Targets(targetList)
                .build();

            CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
                .databaseName(dbName)
                .name(crawlerName)
                .description("Created by the AWS Glue Java API")
                .targets(targets)
                .role(iam)
                .schedule(cron)
                .build();

            glueClient.createCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully created");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateCrawler](#)”中的。

GetCrawler

以下代码示例演示如何使用 GetCrawler。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GetCrawlerRequest;
import software.amazon.awssdk.services.glue.model.GetCrawlerResponse;
import software.amazon.awssdk.services.glue.model.GlueException;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetCrawler {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <crawlerName>

                Where:
                crawlerName - The name of the crawler.\s
    }
}
```

```
        """);

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String crawlerName = args[0];
    Region region = Region.US_EAST_1;
    GlueClient glueClient = GlueClient.builder()
        .region(region)
        .build();

    getSpecificCrawler(glueClient, crawlerName);
    glueClient.close();
}

public static void getSpecificCrawler(GlueClient glueClient, String crawlerName)
{
    try {
        GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        GetCrawlerResponse response = glueClient.getCrawler(crawlerRequest);
        Instant createDate = response.crawler().creationTime();

        // Convert the Instant to readable date
        DateTimeFormatter formatter =
        DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
            .withLocale(Locale.US)
            .withZone(ZoneId.systemDefault());

        formatter.format(createDate);
        System.out.println("The create date of the Crawler is " + createDate);

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetCrawler](#)”中的。

GetDatabase

以下代码示例演示如何使用 GetDatabase。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GetDatabaseRequest;
import software.amazon.awssdk.services.glue.model.GetDatabaseResponse;
import software.amazon.awssdk.services.glue.model.GlueException;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetDatabase {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <databaseName>

                Where:
                databaseName - The name of the database.\s
    }
}
```



```
        """);

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String databaseName = args[0];
    Region region = Region.US_EAST_1;
    GlueClient glueClient = GlueClient.builder()
        .region(region)
        .build();

    getSpecificDatabase(glueClient, databaseName);
    glueClient.close();
}

public static void getSpecificDatabase(GlueClient glueClient, String
databaseName) {
    try {
        GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
            .name(databaseName)
            .build();

        GetDatabaseResponse response = glueClient.getDatabase(databasesRequest);
        Instant createDate = response.database().createTime();

        // Convert the Instant to readable date.
        DateTimeFormatter formatter =
        DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
            .withLocale(Locale.US)
            .withZone(ZoneId.systemDefault());

        formatter.format(createDate);
        System.out.println("The create date of the database is " + createDate);

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetDatabase](#)”中的。

GetTables

以下代码示例演示如何使用 GetTables。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GetTableRequest;
import software.amazon.awssdk.services.glue.model.GetTableResponse;
import software.amazon.awssdk.services.glue.model.GlueException;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetTable {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <dbName> <tableName>

                Where:
                dbName - The database name.\s
```

```
        tableName - The name of the table.\s
        """);

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String dbName = args[0];
    String tableName = args[1];
    Region region = Region.US_EAST_1;
    GlueClient glueClient = GlueClient.builder()
        .region(region)
        .build();

    getGlueTable(glueClient, dbName, tableName);
    glueClient.close();
}

public static void getGlueTable(GlueClient glueClient, String dbName, String
tableName) {
    try {
        GetTableRequest tableRequest = GetTableRequest.builder()
            .databaseName(dbName)
            .name(tableName)
            .build();

        GetTableResponse tableResponse = glueClient.getTable(tableRequest);
        Instant createDate = tableResponse.table().createTime();

        // Convert the Instant to readable date.
        DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
            .withLocale(Locale.US)
            .withZone(ZoneId.systemDefault());

        formatter.format(createDate);
        System.out.println("The create date of the table is " + createDate);

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetTables](#)”中的。

StartCrawler

以下代码示例演示如何使用 StartCrawler。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GlueException;
import software.amazon.awssdk.services.glue.model.StartCrawlerRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class StartCrawler {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <crawlerName>

                Where:
                crawlerName - The name of the crawler.\s
        """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String crawlerName = args[0];
        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
            .region(region)
            .build();

        startSpecificCrawler(glueClient, crawlerName);
        glueClient.close();
    }

    public static void startSpecificCrawler(GlueClient glueClient, String
crawlerName) {
        try {
            StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            glueClient.startCrawler(crawlerRequest);

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartCrawler](#)”中的。

场景

爬网程序和作业入门

以下代码示例展示了如何：

- 创建一个爬虫来抓取公有 Amazon S3 存储桶并生成格式CSV元数据的数据库。
- 列出您的中的数据库和表的相关信息 AWS Glue Data Catalog。

- 创建任务以从 S3 存储桶中提取 CSV 数据、转换数据，并将 JSON 格式化的输出加载到另一个 S3 存储桶中。
- 列出有关作业运行的信息，查看转换后的数据，并清除资源。

有关更多信息，请参阅[教程：AWS Glue Studio 入门](#)。

SDK 适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**
 *
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To set up the resources, see this documentation topic:
 *
 * https://docs.aws.amazon.com/glue/latest/ug/tutorial-add-crawler.html
 *
 * This example performs the following tasks:
 *
 * 1. Create a database.
 * 2. Create a crawler.
 * 3. Get a crawler.
 * 4. Start a crawler.
 * 5. Get a database.
 * 6. Get tables.
 * 7. Create a job.
 * 8. Start a job run.
 * 9. List all jobs.
 * 10. Get job runs.
 * 11. Delete a job.
 * 12. Delete a database.
```

```

* 13. Delete a crawler.
*/

public class GlueScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = ""

            Usage:
                <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName>\s

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

                """;

        if (args.length != 9) {
            System.out.println(usage);
            System.exit(1);
        }

        String iam = args[0];
        String s3Path = args[1];
        String cron = args[2];
        String dbName = args[3];
        String crawlerName = args[4];
        String jobName = args[5];
        String scriptLocation = args[6];
        String locationUri = args[7];
        String bucketNameSc = args[8];

        Region region = Region.US_EAST_1;

```

```
GlueClient glueClient = GlueClient.builder()
    .region(region)
    .build();
System.out.println(DASHES);
System.out.println("Welcome to the AWS Glue scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create a database.");
createDatabase(glueClient, dbName, locationUri);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Create a crawler.");
createGlueCrawler(glueClient, iam, s3Path, cron, dbName, crawlerName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Get a crawler.");
getSpecificCrawler(glueClient, crawlerName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Start a crawler.");
startSpecificCrawler(glueClient, crawlerName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Get a database.");
getSpecificDatabase(glueClient, dbName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("*** Wait 5 min for the tables to become available");
TimeUnit.MINUTES.sleep(5);
System.out.println("6. Get tables.");
String myTableName = getGlueTables(glueClient, dbName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Create a job.");
createJob(glueClient, jobName, iam, scriptLocation);
System.out.println(DASHES);
```



```
        System.out.println(DASHES);
        System.out.println("8. Start a Job run.");
        startJob(glueClient, jobName, dbName, myTableName, bucketNameSc);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. List all jobs.");
        getAllJobs(glueClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get job runs.");
        getJobRuns(glueClient, jobName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete a job.");
        deleteJob(glueClient, jobName);
        System.out.println("*** Wait 5 MIN for the " + crawlerName + " to stop");
        TimeUnit.MINUTES.sleep(5);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Delete a database.");
        deleteDatabase(glueClient, dbName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Delete a crawler.");
        deleteSpecificCrawler(glueClient, crawlerName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Successfully completed the AWS Glue Scenario");
        System.out.println(DASHES);
    }

    public static void createDatabase(GlueClient glueClient, String dbName, String
locationUri) {
        try {
            DatabaseInput input = DatabaseInput.builder()
                .description("Built with the AWS SDK for Java V2")
                .name(dbName)
                .locationUri(locationUri)
```

```
        .build();

        CreateDatabaseRequest request = CreateDatabaseRequest.builder()
            .databaseInput(input)
            .build();

        glueClient.createDatabase(request);
        System.out.println(dbName + " was successfully created");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createGlueCrawler(GlueClient glueClient,
    String iam,
    String s3Path,
    String cron,
    String dbName,
    String crawlerName) {

    try {
        S3Target s3Target = S3Target.builder()
            .path(s3Path)
            .build();

        List<S3Target> targetList = new ArrayList<>();
        targetList.add(s3Target);
        CrawlerTargets targets = CrawlerTargets.builder()
            .s3Targets(targetList)
            .build();

        CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
            .databaseName(dbName)
            .name(crawlerName)
            .description("Created by the AWS Glue Java API")
            .targets(targets)
            .role(iam)
            .schedule(cron)
            .build();

        glueClient.createCrawler(crawlerRequest);
        System.out.println(crawlerName + " was successfully created");
    }
}
```

```
    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getSpecificCrawler(GlueClient glueClient, String crawlerName)
{
    try {
        GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        boolean ready = false;
        while (!ready) {
            GetCrawlerResponse response = glueClient.getCrawler(crawlerRequest);
            String status = response.crawler().stateAsString();
            if (status.compareTo("READY") == 0) {
                ready = true;
            }
            Thread.sleep(3000);
        }

        System.out.println("The crawler is now ready");

    } catch (GlueException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void startSpecificCrawler(GlueClient glueClient, String
crawlerName) {
    try {
        StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        glueClient.startCrawler(crawlerRequest);
        System.out.println(crawlerName + " was successfully started!");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
        System.exit(1);
    }
}

public static void getSpecificDatabase(GlueClient glueClient, String
databaseName) {
    try {
        GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
            .name(databaseName)
            .build();

        GetDatabaseResponse response = glueClient.getDatabase(databasesRequest);
        Instant createDate = response.database().createTime();

        // Convert the Instant to readable date.
        DateTimeFormatter formatter =
        DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
            .withLocale(Locale.US)
            .withZone(ZoneId.systemDefault());

        formatter.format(createDate);
        System.out.println("The create date of the database is " + createDate);

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static String getGlueTables(GlueClient glueClient, String dbName) {
    String myTableName = "";
    try {
        GetTablesRequest tableRequest = GetTablesRequest.builder()
            .databaseName(dbName)
            .build();

        GetTablesResponse response = glueClient.getTables(tableRequest);
        List<Table> tables = response.tableList();
        if (tables.isEmpty()) {
            System.out.println("No tables were returned");
        } else {
            for (Table table : tables) {
                myTableName = table.name();
                System.out.println("Table name is: " + myTableName);
            }
        }
    }
}
```

```
        }
    }

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return myTableName;
}

public static void startJob(GlueClient glueClient, String jobName, String
inputDatabase, String inputTable,
    String outBucket) {
    try {
        Map<String, String> myMap = new HashMap<>();
        myMap.put("--input_database", inputDatabase);
        myMap.put("--input_table", inputTable);
        myMap.put("--output_bucket_url", outBucket);

        StartJobRunRequest runRequest = StartJobRunRequest.builder()
            .workerType(WorkerType.G_1_X)
            .numberOfWorkers(10)
            .arguments(myMap)
            .jobName(jobName)
            .build();

        StartJobRunResponse response = glueClient.startJobRun(runRequest);
        System.out.println("The request Id of the job is " +
response.responseMetadata().requestId());

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createJob(GlueClient glueClient, String jobName, String iam,
String scriptLocation) {
    try {
        JobCommand command = JobCommand.builder()
            .pythonVersion("3")
            .name("glueetl")
            .scriptLocation(scriptLocation)
            .build();
    }
}
```

```
        CreateJobRequest jobRequest = CreateJobRequest.builder()
            .description("A Job created by using the AWS SDK for Java V2")
            .glueVersion("2.0")
            .workerType(WorkerType.G_1_X)
            .numberOfWorkers(10)
            .name(jobName)
            .role(iam)
            .command(command)
            .build();

        glueClient.createJob(jobRequest);
        System.out.println(jobName + " was successfully created.");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getAllJobs(GlueClient glueClient) {
    try {
        GetJobsRequest jobsRequest = GetJobsRequest.builder()
            .maxResults(10)
            .build();

        GetJobsResponse jobsResponse = glueClient.getJobs(jobsRequest);
        List<Job> jobs = jobsResponse.jobs();
        for (Job job : jobs) {
            System.out.println("Job name is : " + job.name());
            System.out.println("The job worker type is : " +
job.workerType().name());
        }

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getJobRuns(GlueClient glueClient, String jobName) {
    try {
        GetJobRunsRequest runsRequest = GetJobRunsRequest.builder()
            .jobName(jobName)
```

```
        .maxResults(20)
        .build();

    boolean jobDone = false;
    while (!jobDone) {
        GetJobRunsResponse response = glueClient.getJobRuns(runsRequest);
        List<JobRun> jobRuns = response.jobRuns();
        for (JobRun jobRun : jobRuns) {
            String jobState = jobRun.jobRunState().name();
            if (jobState.compareTo("SUCCEEDED") == 0) {
                System.out.println(jobName + " has succeeded");
                jobDone = true;

            } else if (jobState.compareTo("STOPPED") == 0) {
                System.out.println("Job run has stopped");
                jobDone = true;

            } else if (jobState.compareTo("FAILED") == 0) {
                System.out.println("Job run has failed");
                jobDone = true;

            } else if (jobState.compareTo("TIMEOUT") == 0) {
                System.out.println("Job run has timed out");
                jobDone = true;

            } else {
                System.out.println("*** Job run state is " +
jobRun.jobRunState().name());
                System.out.println("Job run Id is " + jobRun.id());
                System.out.println("The Glue version is " +
jobRun.glueVersion());
            }
            TimeUnit.SECONDS.sleep(5);
        }
    }

    } catch (GlueException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void deleteJob(GlueClient glueClient, String jobName) {
    try {
```

```
        DeleteJobRequest jobRequest = DeleteJobRequest.builder()
            .jobName(jobName)
            .build();

        glueClient.deleteJob(jobRequest);
        System.out.println(jobName + " was successfully deleted");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteDatabase(GlueClient glueClient, String databaseName) {
    try {
        DeleteDatabaseRequest request = DeleteDatabaseRequest.builder()
            .name(databaseName)
            .build();

        glueClient.deleteDatabase(request);
        System.out.println(databaseName + " was successfully deleted");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteSpecificCrawler(GlueClient glueClient, String
crawlerName) {
    try {
        DeleteCrawlerRequest deleteCrawlerRequest =
DeleteCrawlerRequest.builder()
            .name(crawlerName)
            .build();

        glueClient.deleteCrawler(deleteCrawlerRequest);
        System.out.println(crawlerName + " was deleted");

    } catch (GlueException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```



```
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [CreateCrawler](#)
 - [CreateJob](#)
 - [DeleteCrawler](#)
 - [DeleteDatabase](#)
 - [DeleteJob](#)
 - [DeleteTable](#)
 - [GetCrawler](#)
 - [GetDatabase](#)
 - [GetDatabases](#)
 - [GetJob](#)
 - [GetJobRun](#)
 - [GetJobRuns](#)
 - [GetTables](#)
 - [ListJobs](#)
 - [StartCrawler](#)
 - [StartJobRun](#)

HealthImaging 用SDK于 Java 2.x 的示例

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 HealthImaging。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)
- [场景](#)

操作

CopyImageSet

以下代码示例演示如何使用 CopyImageSet。

SDK适用于 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;
}
}
```

用于创建可复制属性的实用函数。

```

/**
 * 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();
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CopyImageSet](#)”中的。

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

CreateDatastore

以下代码示例演示如何使用 CreateDatastore。

SDK适用于 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 "";
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateDatastore](#)”中的。

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

DeleteDatastore

以下代码示例演示如何使用 DeleteDatastore。

SDK适用于 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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteDatastore](#)”中的。

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

DeleteImageSet

以下代码示例演示如何使用 DeleteImageSet。

SDK适用于 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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteImageSet](#)”中的。

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

GetDICOMImportJob

以下代码示例演示如何使用 GetDICOMImportJob。

SDK适用于 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;
}
```

- 有关API详细信息，请参阅《AWS SDK for Java 2.x API参考资料》中的 [GetDICOMImportJob](#)。

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

GetDatastore

以下代码示例演示如何使用 GetDatastore。

SDK适用于 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;
}
```


- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetDatastore](#)”中的。

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

GetImageFrame

以下代码示例演示如何使用 GetImageFrame。

SDK适用于 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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetImageFrame](#)”中的。

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

GetImageSet

以下代码示例演示如何使用 GetImageSet。

SDK适用于 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;
    }
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetImageSet](#)”中的。

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

GetImageSetMetadata

以下代码示例演示如何使用 `GetImageSetMetadata`。

SDK适用于 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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetImageSetMetadadata](#)”中的。

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

ListDICOMImportJobs

以下代码示例演示如何使用 ListDICOMImportJobs。

SDK适用于 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<>();
}
```

- 有关API详细信息，请参阅AWS SDK for Java 2.x API参考中的 [ListDICOMImport Jobs](#)。

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

ListDatastores

以下代码示例演示如何使用 ListDatastores。

SDK适用于 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;
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListDatastores](#)”中的。

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

ListImageSetVersions

以下代码示例演示如何使用 ListImageSetVersions。

SDK适用于 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;
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListImageSetVersions](#)”中的。

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

ListTagsForResource

以下代码示例演示如何使用 ListTagsForResource。

SDK适用于 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;
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListTagsForResource](#)”中的。

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

SearchImageSets

以下代码示例演示如何使用 SearchImageSets。

SDK适用于 Java 2.x

用于搜索映像集的实用程序函数。

```
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;
}

```

用例 #1: EQUAL 运算符。

```

        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();
    }
}

```

用例 #2: BETWEEN 运算符使用DICOMStudyDate和DICOMStudyTime。

```

DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyyMMdd");
searchFilters = Collections.singletonList(SearchFilter.builder()
    .operator(Operator.BETWEEN)
    .values(SearchByAttributeValue.builder()

.dicomStudyDateAndTime(DICOMStudyDateAndTime.builder()
    .dicomStudyDate("19990101")
    .dicomStudyTime("000000.000")
    .build())
    .build(),
    SearchByAttributeValue.builder()

.dicomStudyDateAndTime(DICOMStudyDateAndTime.builder()
    .dicomStudyDate((LocalDate.now()
        .format(formatter)))
    .dicomStudyTime("000000.000")
    .build())
    .build())
    .build());

searchCriteria = SearchCriteria.builder()
    .filters(searchFilters)
    .build();

imageSetsMetadataSummaries =
searchMedicalImagingImageSets(medicalImagingClient,
    datastoreId, searchCriteria);
if (imageSetsMetadataSummaries != null) {
    System.out.println(
        "The image sets searched with BETWEEN operator using
DICOMStudyDate and DICOMStudyTime are:\n"
        +
        imageSetsMetadataSummaries);
}

```

```

        System.out.println();
    }

```

用例 #3: BETWEEN 运算符使用createdAt。时间研究以前一直存在。

```

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();
}

```

用例 #4: 开启DICOMSeriesInstanceUID和开启EQUAL运算符 updatedAt 并BETWEEN按updatedAt 字段ASC顺序对响应进行排序。

```

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();
        }
    }
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [SearchImageSets](#)”中的。

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

StartDICOMImportJob

以下代码示例演示如何使用 StartDICOMImportJob。

SDK适用于 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 "";
}
```

- 有关API详细信息，请参阅《AWS SDK for Java 2.x API参考资料》中的 [StartDICOMImport Job](#)。

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

TagResource

以下代码示例演示如何使用 TagResource。

SDK适用于 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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [TagResource](#)”中的。

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

UntagResource

以下代码示例演示如何使用 UntagResource。

SDK适用于 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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [UntagResource](#)”中的。

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

UpdateImageSetMetadata

以下代码示例演示如何使用 UpdateImageSetMetadata。

SDK适用于 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;
        }
    }
}

```

用例 #1: 插入或更新属性。

```

final String insertAttributes = ""
    {
        "SchemaVersion": 1.1,
        "Study": {
            "DICOM": {
                "StudyDescription": "CT CHEST"
            }
        }
    }

```

```

        }
    }
    """;
    MetadataUpdates metadataInsertUpdates = MetadataUpdates.builder()
        .dicomUpdates(DICOMUpdates.builder()
            .updateableAttributes(SdkBytes.fromByteBuffer(
                ByteBuffer.wrap(insertAttributes
                    .getBytes(StandardCharsets.UTF_8))))
            .build())
        .build();

    updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
imagesetId,
        versionid, metadataInsertUpdates, force);

```

用例 #2: 移除属性。

```

    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);

```

用例 #3: 移除实例。


```

        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);

```

用例 #4: 恢复到以前的版本。

```

        // 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);

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [UpdateImageSetMetadata](#)”中的。

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

场景

标记数据存储

以下代码示例显示了如何为 HealthImaging 数据存储添加标签。

SDK适用于 Java 2.x

标记数据存储。

```
final String dataStoreArn = "arn:aws:medical-imaging:us-east-1:123456789012:datastore/12345678901234567890123456789012";

TagResource.tagMedicalImagingResource(medicalImagingClient,
dataStoreArn,
ImmutableMap.of("Deployment", "Development"));
```

用于标记资源的实用程序函数。

```
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);
    }
}
```

列出数据存储的标签。

```
        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());
        }
}
```

用于列出资源标签的实用程序函数。

```
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;
}
```

取消标记数据存储。

```
final String datastoreArn = "arn:aws:medical-imaging:us-east-1:123456789012:datastore/12345678901234567890123456789012";

UntagResource.untagMedicalImagingResource(medicalImagingClient,
datastoreArn,
Collections.singletonList("Deployment"));
```

用于取消标记资源的实用程序函数。

```
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);
}
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [ListTagsForResource](#)
 - [TagResource](#)
 - [UntagResource](#)

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

标记映像集

以下代码示例显示了如何为 HealthImaging 图像集添加标签。

SDK适用于 Java 2.x

标记映像集。

```
final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";

TagResource.tagMedicalImagingResource(medicalImagingClient,
imageSetArn,
                                     ImmutableMap.of("Deployment", "Development"));
```

用于标记资源的实用程序函数。

```
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);
    }
}
```

列出映像集的标签。

```
        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());
        }
    }
```

用于列出资源标签的实用程序函数。

```
    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;
    }
}
```

取消标记映像集。


```
        final String imageSetArn = "arn:aws:medical-imaging:us-
east-1:123456789012:datastore/12345678901234567890123456789012/
imageset/12345678901234567890123456789012";
```

```
UntagResource.untagMedicalImagingResource(medicalImagingClient,  
imageSetArn,  
Collections.singletonList("Deployment"));
```

用于取消标记资源的实用程序函数。

```
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);  
    }  
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [ListTagsForResource](#)
 - [TagResource](#)
 - [UntagResource](#)

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

IAM用SDK于 Java 2.x 的示例

以下代码示例向您展示了如何使用with来执行操作和实现常见场景IAM。 AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

开始使用

开始使用 IAM

以下代码示例展示了如何开始使用IAM。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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());
    });
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListPolicies](#)”中的。

主题

- [操作](#)
- [场景](#)

操作

AttachRolePolicy

以下代码示例演示如何使用 AttachRolePolicy。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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");
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [AttachRolePolicy](#)”中的。

CreateAccessKey

以下代码示例演示如何使用 CreateAccessKey。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateAccessKey](#)”中的。

CreateAccountAlias

以下代码示例演示如何使用 CreateAccountAlias。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateAccountAlias](#)”中的。

CreatePolicy

以下代码示例演示如何使用 CreatePolicy。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";
}
```

```
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreatePolicy](#)”中的。

CreateRole

以下代码示例演示如何使用 CreateRole。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateRole](#)”中的。

CreateUser

以下代码示例演示如何使用 CreateUser。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateUser](#)”中的。

DeleteAccessKey

以下代码示例演示如何使用 DeleteAccessKey。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteAccessKey](#)”中的。

DeleteAccountAlias

以下代码示例演示如何使用 DeleteAccountAlias。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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");
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteAccountAlias](#)”中的。

DeletePolicy

以下代码示例演示如何使用 DeletePolicy。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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");
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeletePolicy](#)”中的。

DeleteUser

以下代码示例演示如何使用 DeleteUser。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteUser](#)”中的。

DetachRolePolicy

以下代码示例演示如何使用 DetachRolePolicy。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DetachRolePolicy](#)”中的。

ListAccessKeys

以下代码示例演示如何使用 ListAccessKeys。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListAccessKeys](#)”中的。

ListAccountAliases

以下代码示例演示如何使用 ListAccountAliases。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListAccountAliases](#)”中的。

ListUsers

以下代码示例演示如何使用 ListUsers。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
}
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListUsers](#)”中的。

UpdateAccessKey

以下代码示例演示如何使用 UpdateAccessKey。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [UpdateAccessKey](#)”中的。

UpdateUser

以下代码示例演示如何使用 UpdateUser。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [UpdateUser](#)”中的。

场景

构建和管理弹性服务

以下代码示例演示了如何创建可返回书籍、电影和歌曲推荐的负载均衡的 Web 服务。该示例演示服务如何响应故障，以及如何重组服务以提高故障发生时的弹性。

- 使用 Amazon A EC2 uto Scaling 组根据启动模板创建亚马逊弹性计算云 (AmazonEC2) 实例，并将实例数量保持在指定范围内。
- 使用 Elastic Load Balancing 处理和分发HTTP请求。
- 监控自动扩缩组中实例的运行状况，并仅将请求转发到运行状况良好的实例。
- 在每个EC2实例上运行 Python 网络服务器来处理HTTP请求。Web 服务器以建议和运行状况检查作为响应。
- 使用 Amazon DynamoDB 表模拟推荐服务。
- 通过更新 AWS Systems Manager 参数来控制 Web 服务器对请求和运行状况检查的响应。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，了解如何进行设置和运行。

在命令提示符中运行交互式场景。

```
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 wasSuccessful = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessful) {
            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 (wasSuccessful) {
    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("\n0kay, 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);
}
}

```

创建一个包含 Auto Scaling 和 Amazon EC2 操作的类。

```

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
    .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.getInstanceProfile().getInstanceProfileName();
        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.");
    }
}

```

创建一个包含弹性负载均衡操作的类。

```

public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                .region(Region.US_EAST_1)
                .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
            .names(targetGroupName)
            .build();

        DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()
            .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
            .build();

        DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }
}

```

```
}

// Gets the HTTP endpoint of the load balancer.
public String getEndpoint(String lbName) {
    DescribeLoadBalancersResponse res = getLoadBalancerClient()
        .describeLoadBalancers(describe -> describe.names(lbName));
    return res.loadBalancers().get(0).dnsName();
}

// Deletes a load balancer.
public void deleteLoadBalancer(String lbName) {
    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
            .loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
            .build();

        getLoadBalancerClient().deleteLoadBalancer(
            builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancersDeleted(request);
        waiterResponse.matched().response().ifPresent(System.out::println);

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}

// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
            .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
```

```

        .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}

// Verify this computer can successfully send a GET request to the load balancer
// endpoint.
public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws IOException,
InterruptedException {
    boolean success = false;
    int retries = 3;
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to the ELB.
    HttpGet httpGet = new HttpGet("http://" + elbDnsName);
    try {
        while ((!success) && (retries > 0)) {
            // Execute the request and get the response.
            HttpResponse response = httpClient.execute(httpGet);
            int statusCode = response.getStatusLine().getStatusCode();
            System.out.println("HTTP Status Code: " + statusCode);
            if (statusCode == 200) {
                success = true;
            } else {
                retries--;
                System.out.println("Got connection error from load balancer
endpoint, retrying...");
                TimeUnit.SECONDS.sleep(15);
            }
        }

    } catch (org.apache.http.conn.HttpHostConnectException e) {
        System.out.println(e.getMessage());
    }

    System.out.println("Status.." + success);
    return success;
}

/*
 * Creates an Elastic Load Balancing target group. The target group specifies

```

```
    * how
    * the load balancer forward requests to instances in the group and how instance
    * health is checked.
    */
    public String createTargetGroup(String protocol, int port, String vpcId, String
targetGroupName) {
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
            .healthCheckPath("/healthcheck")
            .healthCheckTimeoutSeconds(5)
            .port(port)
            .vpcId(vpcId)
            .name(targetGroupName)
            .protocol(protocol)
            .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /**
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
        String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                .map(Subnet::subnetId)
                .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                .subnets(subnetIdStrings)
                .name(lbName)
```

```
        .scheme("internet-facing")
        .build();

    // Create and wait for the load balancer to become available.
    CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
    String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

    ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
    DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
        .loadBalancerArns(lbARN)
        .build();

    System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
    WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
        .waitUntilLoadBalancerAvailable(request);
    waiterResponse.matched().response().ifPresent(System.out::println);
    System.out.println("Load Balancer " + lbName + " is available.");

    // Get the DNS name (endpoint) of the load balancer.
    String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
    System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

    // Create a listener for the load balance.
    Action action = Action.builder()
        .targetGroupArn(targetGroupARN)
        .type("forward")
        .build();

    CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
        .defaultActions(action)
        .port(port)
        .protocol(protocol)
        .defaultActions(action)
        .build();

    getLoadBalancerClient().createListener(listenerRequest);
```

```

        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
            + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
}

```

创建一个使用 DynamoDB 模拟推荐服务的类。

```

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);
}
}
```

创建一个包含 Systems Manager 操作的类。

```
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);
    }
}
```

• 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。

- [AttachLoadBalancerTargetGroups](#)
- [CreateAutoScalingGroup](#)
- [CreateInstanceProfile](#)
- [CreateLaunchTemplate](#)
- [CreateListener](#)
- [CreateLoadBalancer](#)
- [CreateTargetGroup](#)
- [DeleteAutoScalingGroup](#)
- [DeleteInstanceProfile](#)
- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeIamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplacelamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

创建用户并代入角色

以下代码示例展示了如何创建用户并代入角色。

Warning

为避免安全风险，在开发专用软件或处理真实数据时，请勿使用IAM用户进行身份验证。而是使用与身份提供商的联合身份验证，例如 [AWS IAM Identity Center](#)。

- 创建没有权限的用户。
- 创建授予列出账户的 Amazon S3 存储桶的权限的角色
- 添加策略以允许用户代入该角色。
- 代入角色并使用临时凭证列出 S3 存储桶，然后清除资源。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

创建封装IAM用户操作的函数。

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

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。

- [AttachRolePolicy](#)
- [CreateAccessKey](#)
- [CreatePolicy](#)
- [CreateRole](#)
- [CreateUser](#)
- [DeleteAccessKey](#)
- [DeletePolicy](#)
- [DeleteRole](#)
- [DeleteUser](#)
- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

使用IAM策略生成器 API

以下代码示例展示了如何：

- 使用面向对象API创建IAM策略。
- 将IAM策略生成器API与IAM服务配合使用。

SDK适用于 Java 2.x

Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

示例使用以下导入。

```
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;
```

创建基于时间的策略。

```
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());
}

```

创建具有多个条件的策略。

```

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());

}

```

在策略中使用主体。

```

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());
    }

```

允许跨账户存取。

```

    public String allowCrossAccountAccessExample() {
        IamPolicy policy = IamPolicy.builder()
            .addStatement(b -> b
                .effect(IamEffect.ALLOW)
                .addPrincipal(IamPrincipalType.AWS,
                    "111122223333")
                .addAction("s3:PutObject")
                .addResource("arn:aws:s3::DOC-
                    EXAMPLE-BUCKET/*")
                .addCondition(b1 -> b1
                    .operator(IamConditionOperator.STRING_EQUALS)
                    .key("s3:x-amz-acl")
                    .value("bucket-
                    owner-full-control")))
            .build();
        return policy.toJson(IamPolicyWriter.builder()
            .prettyPrint(true).build());
    }

```

生成并上传 IamPolicy。

```

    public String createAndUploadPolicyExample(IamClient iam, String accountID,
        String policyName) {
        // Build the policy.
        IamPolicy policy = IamPolicy.builder() // 'version' defaults to
            "2012-10-17".
            .addStatement(IamStatement.builder()
                .effect(IamEffect.ALLOW)
                .addAction("dynamodb:PutItem")
                .addResource("arn:aws:dynamodb:us-
                    east-1:" + accountID

```

```

                                                                    + ":table/
exampleTableName")
                                                                    .build())
                                                                    .build();
    // Upload the policy.
    iam.createPolicy(r ->
r.policyName(policyName).policyDocument(policy.toJson()));
    return
policy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
}

```

下载并使用 `IamPolicy`。

```

    public String createNewBasedOnExistingPolicyExample(IamClient iam, String
accountID, String policyName,
                String newPolicyName) {

        String policyArn = "arn:aws:iam::" + accountID + ":policy/" +
policyName;
        GetPolicyResponse getPolicyResponse = iam.getPolicy(r ->
r.policyArn(policyArn));

        String policyVersion =
getPolicyResponse.policy().defaultVersionId();
        GetPolicyVersionResponse getPolicyVersionResponse = iam
                .getPolicyVersion(r ->
r.policyArn(policyArn).versionId(policyVersion));

        // Create an IamPolicy instance from the JSON string returned from
IAM.
        String decodedPolicy =
URLDecoder.decode(getPolicyVersionResponse.policyVersion().document(),
                StandardCharsets.UTF_8);
        IamPolicy policy = IamPolicy.fromJson(decodedPolicy);

        /*
        * All IamPolicy components are immutable, so use the copy method
that creates a
        * new instance that
        * can be altered in the same method call.
        */

```

```
        * Add the ability to get an item from DynamoDB as an additional
action.
        */
        IAMStatement newStatement = policy.statements().get(0).copy(s ->
s.addAction("dynamodb:GetItem"));

        // Create a new statement that replaces the original statement.
        IAMPolicy newPolicy = policy.copy(p ->
p.statements(Arrays.asList(newStatement)));

        // Upload the new policy. IAM now has both policies.
        iam.createPolicy(r -> r.policyName(newPolicyName)
            .policyDocument(newPolicy.toJson()));

        return
newPolicy.toJson(IAMPolicyWriter.builder().prettyPrint(true).build());
    }
}
```

- 有关更多信息，请参阅 [AWS SDK for Java 2.x 开发人员指南](#)。
- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [CreatePolicy](#)
 - [GetPolicy](#)
 - [GetPolicyVersion](#)

AWS IoT 用SDK于 Java 2.x 的示例

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS IoT。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

开始使用

你好 AWS IoT

以下代码示例展示了如何开始使用 AWS IoT。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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());
            });
    }
}
```

```
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [listThings](#)”中的。

主题

- [操作](#)
- [场景](#)

操作

AttachThingPrincipal

以下代码示例演示如何使用 `AttachThingPrincipal`。

SDK适用于 Java 2.x

Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**
 * 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();
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [AttachThingPrincipal](#)”中的。

CreateKeysAndCertificate

以下代码示例演示如何使用 CreateKeysAndCertificate。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```

/**
 * 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];
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateKeysAndCertificate](#)”中的。

CreateThing

以下代码示例演示如何使用 CreateThing。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**
 * 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();  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateThing](#)”中的。

CreateTopicRule

以下代码示例演示如何使用 CreateTopicRule。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**  
 * 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();
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateTopicRule](#)”中的。

DeleteCertificate

以下代码示例演示如何使用 DeleteCertificate。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**
 * 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();
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteCertificate](#)”中的。

DeleteThing

以下代码示例演示如何使用 DeleteThing。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**
 * 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();  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteThing](#)”中的。

DescribeEndpoint

以下代码示例演示如何使用 DescribeEndpoint。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**  
 * 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().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];
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeEndpoint](#)”中的。

DescribeThing

以下代码示例演示如何使用 DescribeThing。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**
 * 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();
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeThing](#)”中的。

DetachThingPrincipal

以下代码示例演示如何使用 DetachThingPrincipal。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**
 * 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();
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DetachThingPrincipal](#)”中的。

ListCertificates

以下代码示例演示如何使用 ListCertificates。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**
 * 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();
}
```

```
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListCertificates](#)”中的。

SearchIndex

以下代码示例演示如何使用 SearchIndex。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**
 * 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();
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [SearchIndex](#)”中的。

UpdateThing

以下代码示例演示如何使用 UpdateThing。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```

/**
 * 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();
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [UpdateThing](#)”中的。

场景

处理设备管理用例

以下代码示例说明如何使用 AWS IoT 设备管理用例 AWS IoT SDK

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

运行演示 AWS IoT 功能的交互式场景。

```
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 and 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方法的包装器类。

```
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)

                .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
                .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)

.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .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().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);
}
}
```

AWS IoT data 用SDK于 Java 2.x 的示例

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS IoT data。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

GetThingShadow

以下代码示例演示如何使用 GetThingShadow。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**
 * 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();
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetThingShadow](#)”中的。

UpdateThingShadow

以下代码示例演示如何使用 UpdateThingShadow。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**
 * 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();
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [UpdateThingShadow](#)”中的。

SDK适用于 Java 2.x 的 Amazon Keyspaces 示例

以下代码示例向您展示了如何使用 AWS SDK for Java 2.x 与 Amazon Keyspaces 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

开始使用

Hello Amazon Keysp

以下代码示例展示了如何开始使用 Amazon Keyspaces。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

```
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListKeyspaces](#)”中的。

主题

- [操作](#)
- [场景](#)

操作

CreateKeyspace

以下代码示例演示如何使用 CreateKeyspace。

SDK适用于 Java 2.x

Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```


- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateKeyspace](#)”中的。

CreateTable

以下代码示例演示如何使用 CreateTable。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
```

```
collList.add(defYear);
collList.add(defReleaseDate);
collList.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(collList)
    .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);
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateTable](#)”中的。

DeleteKeyspace

以下代码示例演示如何使用 DeleteKeyspace。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteKeyspace](#)”中的。

DeleteTable

以下代码示例演示如何使用 DeleteTable。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteTable](#)”中的。

GetKeyspace

以下代码示例演示如何使用 GetKeyspace。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetKeyspace](#)”中的。

GetTable

以下代码示例演示如何使用 GetTable。

SDK适用于 Java 2.x

Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetTable](#)”中的。

ListKeyspaces

以下代码示例演示如何使用 ListKeyspaces。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListKeyspaces](#)”中的。

ListTables

以下代码示例演示如何使用 ListTables。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListTables](#)”中的。

RestoreTable

以下代码示例演示如何使用 RestoreTable。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```

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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [RestoreTable](#)”中的。

UpdateTable

以下代码示例演示如何使用 UpdateTable。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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)
            .addColumnns(def)
            .build();

        keyClient.updateTable(tableRequest);
    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [UpdateTable](#)”中的。

场景

开始使用密钥空间和表

以下代码示例演示了操作流程：

- 创建密钥空间和表。表架构保存电影数据并启用了 point-in-time 恢复。
- 使用带有 Sigv4 身份验证的安全TLS连接连接到密钥空间。
- 查询表。添加、检索和更新电影数据。
- 更新表。添加一系列来跟踪观看的电影。
- 将表还原到以前的状态并清理资源。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including 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)
                .addColumnns(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);
    }
}
```

```
}  
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [CreateKeyspace](#)
 - [CreateTable](#)
 - [DeleteKeyspace](#)
 - [DeleteTable](#)
 - [GetKeyspace](#)
 - [GetTable](#)
 - [ListKeyspaces](#)
 - [ListTables](#)
 - [RestoreTable](#)
 - [UpdateTable](#)

使用SDK适用于 Java 2.x 的 Kinesis 示例

以下代码示例向您展示了如何使用 AWS SDK for Java 2.x 与 Kinesis 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)
- [无服务器示例](#)

操作

CreateStream

以下代码示例演示如何使用 CreateStream。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateStream](#)”中的。

DeleteStream

以下代码示例演示如何使用 DeleteStream。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteStream](#)”中的。

GetRecords

以下代码示例演示如何使用 GetRecords。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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()));
```

```
    }  
  }  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetRecords](#)”中的。

PutRecord

以下代码示例演示如何使用 PutRecord。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [PutRecord](#)”中的。

无服务器示例

通过 Kinesis 触发器调用 Lambda 函数

以下代码示例展示了如何实现一个 Lambda 函数，该函数接收因接收来自 Kinesis 流的记录而触发的事件。该函数检索 Kinesis 有效负载，将 Base64 解码，并记录下记录内容。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

使用 Java 将 Kinesis 事件与 Lambda 结合使用。

```
// 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;
    }
}
```

通过 Kinesis 触发器报告 Lambda 函数批处理项目失败

以下代码示例展示了如何为接收来自 Kinesis 流的事件的 Lambda 函数实现部分批处理响应。该函数在响应中报告批处理项目失败，并指示 Lambda 稍后重试这些消息。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

报告使用 Java 进行 Lambda Kinesis 批处理项目失败。

```
// 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 用SDK于 Java 2.x 的示例

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS KMS。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

开始使用

你好KMS钥匙

以下代码示例显示了如何开始使用 KMS key。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.paginators.ListKeysIterable;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class HelloKMS {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
            .region(region)
            .build();

        listAllKeys(kmsClient);
        kmsClient.close();
    }

    public static void listAllKeys(KmsClient kmsClient) {
        try {
            ListKeysRequest listKeysRequest = ListKeysRequest.builder()
                .limit(15)
                .build();

            ListKeysIterable keysResponse =
kmsClient.listKeysPaginator(listKeysRequest);
            keysResponse.stream()
                .flatMap(r -> r.keys().stream())
                .forEach(key -> System.out
                    .println(" The key ARN is: " + key.keyArn() + ". The key Id is:
" + key.keyId()));

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [listKeysPaginator](#)”中的。

主题

- [操作](#)
- [场景](#)

操作

CreateAlias

以下代码示例演示如何使用 CreateAlias。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void createCustomAlias(KmsClient kmsClient, String targetKeyId,
String aliasName) {
    try {
        CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
            .aliasName(aliasName)
            .targetKeyId(targetKeyId)
            .build();


        kmsClient.createAlias(aliasRequest);
        System.out.println(aliasName + " was successfully created.");
    } catch (ResourceExistsException e) {
        System.err.println("Alias already exists: " + e.getMessage());
        System.err.println("Moving on...");
    } catch (Exception e) {
        System.err.println("An unexpected error occurred: " + e.getMessage());
        System.err.println("Moving on...");
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateAlias](#)”中的。

CreateGrant

以下代码示例演示如何使用 CreateGrant。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static String grantKey(KmsClient kmsClient, String keyId, String
granteePrincipal) {
    try {
        // Add the desired KMS Grant permissions.
        List<GrantOperation> grantPermissions = new ArrayList<>();
        grantPermissions.add(GrantOperation.ENCRYPT);
        grantPermissions.add(GrantOperation.DECRYPT);
        grantPermissions.add(GrantOperation.DESCRIBE_KEY);

        CreateGrantRequest grantRequest = CreateGrantRequest.builder()
            .keyId(keyId)
            .name("grant1")
            .granteePrincipal(granteePrincipal)
            .operations(grantPermissions)
            .build();

        CreateGrantResponse response = kmsClient.createGrant(grantRequest);
        return response.grantId();


    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateGrant](#)”中的。

CreateKey

以下代码示例演示如何使用 CreateKey。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static String createKey(KmsClient kmsClient, String keyDesc) {
    try {
        CreateKeyRequest keyRequest = CreateKeyRequest.builder()
            .description(keyDesc)
            .customerMasterKeySpec(CustomerMasterKeySpec.SYMMETRIC_DEFAULT)
            .keyUsage("ENCRYPT_DECRYPT")
            .build();

        CreateKeyResponse result = kmsClient.createKey(keyRequest);
        System.out.println("Symmetric key with ARN [" +
result.keyMetadata().arn() + "] has been created.");
        return result.keyMetadata().keyId();


    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateKey](#)”中的。

Decrypt

以下代码示例演示如何使用 Decrypt。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static String decryptData(KmsClient kmsClient, SdkBytes encryptedData,
String keyId) {
    try {
        DecryptRequest decryptRequest = DecryptRequest.builder()
            .ciphertextBlob(encryptedData)
            .keyId(keyId)
            .build();

        DecryptResponse decryptResponse = kmsClient.decrypt(decryptRequest);
        return decryptResponse.plaintext().asString(StandardCharsets.UTF_8);


    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- 有关API详细信息，请参阅《参考资料》中的“[解密](#)”。AWS SDK for Java 2.x API

DeleteAlias

以下代码示例演示如何使用 DeleteAlias。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void deleteSpecificAlias(KmsClient kmsClient, String aliasName) {
    try {
        DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
            .aliasName(aliasName)
            .build();

        kmsClient.deleteAlias(deleteAliasRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteAlias](#)”中的。

DescribeKey

以下代码示例演示如何使用 DescribeKey。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static boolean isKeyEnabled(KmsClient kmsClient, String keyId) {
    try {
        DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
            .keyId(keyId)
            .build();

        DescribeKeyResponse response = kmsClient.describeKey(keyRequest);
        KeyState keyState = response.keyMetadata().keyState();
        if (keyState == KeyState.ENABLED) {
            System.out.println("The key is enabled.");
            return true;
        } else {
```

```
        System.out.println("The key is not enabled. Key state: " +
keyState);
    }

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return false;
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeKey](#)”中的。

DisableKey

以下代码示例演示如何使用 `DisableKey`。

SDK适用于 Java 2.x

Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void disableKey(KmsClient kmsClient, String keyId) {
    try {
        DisableKeyRequest keyRequest = DisableKeyRequest.builder()
            .keyId(keyId)
            .build();

        kmsClient.disableKey(keyRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DisableKey](#)”中的。

EnableKey

以下代码示例演示如何使用 EnableKey。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
// Enable the KMS key.
public static void enableKey(KmsClient kmsClient, String keyId) {
    try {
        EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
            .keyId(keyId)
            .build();

        kmsClient.enableKey(enableKeyRequest);


    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [EnableKey](#)”中的。

Encrypt

以下代码示例演示如何使用 Encrypt。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static SdkBytes encryptData(KmsClient kmsClient, String keyId, String
text) {
    try {
        SdkBytes myBytes = SdkBytes.fromUtf8String(text);
        EncryptRequest encryptRequest = EncryptRequest.builder()
            .keyId(keyId)
            .plaintext(myBytes)
            .build();

        EncryptResponse response = kmsClient.encrypt(encryptRequest);
        String algorithm = response.encryptionAlgorithm().toString();
        System.out.println("The string was encrypted with algorithm " +
algorithm + ".");

        // Get the encrypted data.
        SdkBytes encryptedData = response.ciphertextBlob();
        return encryptedData;

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return null;
}
```

- 有关API详细信息，请参阅AWS SDK for Java 2.x API参考中的[加密](#)。

ListAliases

以下代码示例演示如何使用 ListAliases。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void listAllAliases(KmsClient kmsClient) {
    try {
        ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
            .limit(15)
            .build();

        ListAliasesIterable aliasesResponse =
kmsClient.listAliasesPaginator(aliasesRequest);
        aliasesResponse.stream()
            .flatMap(r -> r.aliases().stream())
            .forEach(alias -> System.out
                .println("The alias name is: " + alias.aliasName()));

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListAliases](#)”中的。

ListGrants

以下代码示例演示如何使用 ListGrants。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void displayGrantIds(KmsClient kmsClient, String keyId) {
    try {
        ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
            .keyId(keyId)
            .limit(15)
            .build();

        ListGrantsIterable response =
kmsClient.listGrantsPaginator(grantsRequest);
        response.stream()
            .flatMap(r -> r.grants().stream())
            .forEach(grant -> {
                System.out.println("The grant Id is : " + grant.grantId());
                List<GrantOperation> ops = grant.operations();
                for (GrantOperation op : ops) {
                    System.out.println(op.name());
                }
            });
    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListGrants](#)”中的。

ListKeyPolicies

以下代码示例演示如何使用 ListKeyPolicies。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。


```
public static void getKeyPolicy(KmsClient kmsClient, String keyId, String
policyName) {
    try {
        GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
            .keyId(keyId)
            .policyName(policyName)
            .build();

        GetKeyPolicyResponse response = kmsClient.getKeyPolicy(policyRequest);
        System.out.println("The response is "+response.policy());
    } catch (KmsException e) {
        if (e.getMessage().contains("No such policy exists")) {
            System.out.println("The policy cannot be found. Error message: " +
e.getMessage());
        } else {
            throw e;
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListKeyPolicies](#)”中的。

ListKeys

以下代码示例演示如何使用 ListKeys。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.paginators.ListKeysIterable;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloKMS {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        KmsClient kmsClient = KmsClient.builder()
            .region(region)
            .build();

        listAllKeys(kmsClient);
        kmsClient.close();
    }

    public static void listAllKeys(KmsClient kmsClient) {
        try {
            ListKeysRequest listKeysRequest = ListKeysRequest.builder()
                .limit(15)
                .build();

            ListKeysIterable keysResponse =
kmsClient.listKeysPaginator(listKeysRequest);
            keysResponse.stream()
                .flatMap(r -> r.keys().stream())
                .forEach(key -> System.out
                    .println(" The key ARN is: " + key.keyArn() + ". The key Id is:
" + key.keyId()));

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListKeys](#)”中的。

RevokeGrant

以下代码示例演示如何使用 RevokeGrant。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void revokeKeyGrant(KmsClient kmsClient, String keyId, String
grantId) {
    try {
        RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
            .keyId(keyId)
            .grantId(grantId)
            .build();

        kmsClient.revokeGrant(grantRequest);
        System.out.println("Grant ID: [" + grantId + "] was successfully
revoked!");
    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [RevokeGrant](#)”中的。

ScheduleKeyDeletion

以下代码示例演示如何使用 ScheduleKeyDeletion。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void deleteKey(KmsClient kmsClient, String keyId) {
    try {
        ScheduleKeyDeletionRequest deletionRequest =
ScheduleKeyDeletionRequest.builder()
            .keyId(keyId)
            .pendingWindowInDays(7)
            .build();

        kmsClient.scheduleKeyDeletion(deletionRequest);
        System.out.println("The key will be deleted in 7 days.");

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ScheduleKeyDeletion](#)”中的。

Sign

以下代码示例演示如何使用 Sign。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void signVerifyData(KmsClient kmsClient) {
    String signMessage = "Here is the message that will be digitally signed";

    // Create an AWS KMS key used to digitally sign data.
    CreateKeyRequest request = CreateKeyRequest.builder()
        .keySpec(KeySpec.RSA_2048) // Specify key spec
        .keyUsage(KeyUsageType.SIGN_VERIFY) // Specify key usage
        .origin(OriginType.AWS_KMS) // Specify key origin
        .build();

    CreateKeyResponse response = kmsClient.createKey(request);
    String keyId2 = response.keyMetadata().keyId();
    System.out.println("Created KMS key with ID: " + keyId2);

    SdkBytes bytes = SdkBytes.fromString(signMessage, Charset.defaultCharset());
    SignRequest signRequest = SignRequest.builder()
        .keyId(keyId2)
        .message(bytes)
        .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
        .build();

    SignResponse signResponse = kmsClient.sign(signRequest);
    byte[] signedBytes = signResponse.signature().asByteArray();

    // Verify the digital signature.
    VerifyRequest verifyRequest = VerifyRequest.builder()
        .keyId(keyId2)

        .message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset())))
        .signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))
        .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
        .build();

    VerifyResponse verifyResponse = kmsClient.verify(verifyRequest);
    System.out.println("Signature verification result: " +
verifyResponse.signatureValid());
}
```

- 有关API详细信息，请参阅[登录AWS SDK for Java 2.xAPI参考](#)。

TagResource

以下代码示例演示如何使用 TagResource。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void tagKMSKey(KmsClient kmsClient, String keyId) {
    try {
        Tag tag = Tag.builder()
            .tagKey("Environment")
            .tagValue("Production")
            .build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .keyId(keyId)
            .tags(tag)
            .build();

        kmsClient.tagResource(tagResourceRequest);
        System.out.println("The key has been tagged.");
    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [TagResource](#)”中的。

场景

了解KMS关键的核心运营

以下代码示例展示了如何：

- 创建 KMS 密钥。
- 列出您账户的KMS密钥并获取有关密钥的详细信息。
- 启用和禁用KMS密钥。
- 生成可用于客户端加密的对称数据密钥。
- 生成用于对数据进行数字签名的非对称密钥。
- 标记密钥。
- 删除KMS密钥。

SDK适用于 Java 2.x

Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.kms.model.AliasListEntry;
import software.amazon.awssdk.services.kms.model.AlreadyExistsException;
import software.amazon.awssdk.services.kms.model.CreateAliasRequest;
import software.amazon.awssdk.services.kms.model.CreateGrantRequest;
import software.amazon.awssdk.services.kms.model.CreateGrantResponse;
import software.amazon.awssdk.services.kms.model.CreateKeyRequest;
import software.amazon.awssdk.services.kms.model.CreateKeyResponse;
import software.amazon.awssdk.services.kms.model.CustomerMasterKeySpec;
import software.amazon.awssdk.services.kms.model.DecryptRequest;
import software.amazon.awssdk.services.kms.model.DecryptResponse;
import software.amazon.awssdk.services.kms.model.DeleteAliasRequest;
import software.amazon.awssdk.services.kms.model.DescribeKeyRequest;
import software.amazon.awssdk.services.kms.model.DescribeKeyResponse;
import software.amazon.awssdk.services.kms.model.DisableKeyRequest;
import software.amazon.awssdk.services.kms.model.EnableKeyRequest;
import software.amazon.awssdk.services.kms.model.EnableKeyRotationRequest;
import software.amazon.awssdk.services.kms.model.EncryptRequest;
import software.amazon.awssdk.services.kms.model.EncryptResponse;
import software.amazon.awssdk.services.kms.model.GetKeyPolicyRequest;
import software.amazon.awssdk.services.kms.model.GetKeyPolicyResponse;
```

```
import software.amazon.awssdk.services.kms.model.GrantOperation;
import software.amazon.awssdk.services.kms.model.KeySpec;
import software.amazon.awssdk.services.kms.model.KeyState;
import software.amazon.awssdk.services.kms.model.KeyUsageType;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.model.LimitExceededException;
import software.amazon.awssdk.services.kms.model.ListAliasesRequest;
import software.amazon.awssdk.services.kms.model.ListGrantsRequest;
import software.amazon.awssdk.services.kms.model.ListKeyPoliciesRequest;
import software.amazon.awssdk.services.kms.model.ListKeyPoliciesResponse;
import software.amazon.awssdk.services.kms.model.OriginType;
import software.amazon.awssdk.services.kms.model.PutKeyPolicyRequest;
import software.amazon.awssdk.services.kms.model.RevokeGrantRequest;
import software.amazon.awssdk.services.kms.model.ScheduleKeyDeletionRequest;
import software.amazon.awssdk.services.kms.model.SignRequest;
import software.amazon.awssdk.services.kms.model.SignResponse;
import software.amazon.awssdk.services.kms.model.SigningAlgorithmSpec;
import software.amazon.awssdk.services.kms.model.Tag;
import software.amazon.awssdk.services.kms.model.TagResourceRequest;
import software.amazon.awssdk.services.kms.model.VerifyRequest;
import software.amazon.awssdk.services.kms.model.VerifyResponse;
import software.amazon.awssdk.services.kms.paginators.ListAliasesIterable;
import software.amazon.awssdk.services.kms.paginators.ListGrantsIterable;
import software.amazon.awssdk.services.secretsmanager.model.ResourceExistsException;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.services.sts.model.GetCallerIdentityResponse;
import java.nio.ByteBuffer;
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.util.List;
import java.util.ArrayList;
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class KMSScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
```



```
private static final String accountId = getAccountId();

public static void main(String[] args) {
    final String usage = ""
        Usage: <granteePrincipal>

        Where:
            granteePrincipal - The principal (user, service account, or
group) to whom the grant or permission is being given.
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }
    String granteePrincipal = args[0];
    String policyName = "default";

    Scanner scanner = new Scanner(System.in);
    String keyDesc = "Created by the AWS KMS API";

    Region region = Region.US_WEST_2;
    KmsClient kmsClient = KmsClient.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("""
        Welcome to the AWS Key Management SDK Getting Started scenario.

        This program demonstrates how to interact with AWS Key Management using
the AWS SDK for Java (v2).
        The AWS Key Management Service (KMS) is a secure and highly available
service that allows you to create
            and manage AWS KMS keys and control their use across a wide range of AWS
services and applications.
        KMS provides a centralized and unified approach to managing encryption
keys, making it easier to meet your
            data protection and regulatory compliance requirements.

        This Getting Started scenario creates two key types. A symmetric
encryption key is used to encrypt and decrypt data,
            and an asymmetric key used to digitally sign data.
        Let's get started...
    """);
}
```

```

        "");
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("1. Create a symmetric KMS key\n");
        System.out.println("First, the program will creates a symmetric KMS key that
you can used to encrypt and decrypt data.");
        waitForInputToContinue(scanner);
        String targetKeyId = createKey(kmsClient, keyDesc);
        waitForInputToContinue(scanner);

```

```

        System.out.println(DASHES);
        System.out.println("""
            2. Enable a KMS key

```

By default, when the SDK creates an AWS key it is enabled. The next bit of code checks to determine if the key is enabled. If it is not enabled, the code enables it.

```

        "");
        waitForInputToContinue(scanner);
        boolean isEnabled = isKeyEnabled(kmsClient, targetKeyId);
        if (!isEnabled)
            enableKey(kmsClient, targetKeyId);
        waitForInputToContinue(scanner);

```

```

        System.out.println(DASHES);
        System.out.println("3. Encrypt data using the symmetric KMS key");
        String plaintext = "Hello, AWS KMS!";
        System.out.printf("""

```

One of the main uses of symmetric keys is to encrypt and decrypt data.

Next, the code encrypts the string '%s' with the SYMMETRIC_DEFAULT encryption algorithm.

```

            %n""", plaintext);
        waitForInputToContinue(scanner);
        SdkBytes ciphertext = encryptData(kmsClient, targetKeyId, plaintext);
        waitForInputToContinue(scanner);

```

```

        System.out.println(DASHES);
        System.out.println("4. Create an alias");
        System.out.println("""

```

```

        Enter an alias name for the key. The name should be prefixed with
        'alias/'.
        For example, 'alias/myFirstKey'.
        """);

        String aliasName = scanner.nextLine();
        String fullAliasName = aliasName.isEmpty() ? "alias/dev-encryption-key" :
aliasName;
        createCustomAlias(kmsClient, targetKeyId, fullAliasName);
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("5. List all of your aliases");
        waitForInputToContinue(scanner);
        listAllAliases(kmsClient);
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("6. Enable automatic rotation of the KMS key");
        System.out.println("""

                By default, when the SDK enables automatic rotation of a KMS key,
                KMS rotates the key material of the KMS key one year (approximately 365
                days) from the enable date and every year
                thereafter.
                """);
        waitForInputToContinue(scanner);
        enableKeyRotation(kmsClient, targetKeyId);
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
                7. Create a grant

                A grant is a policy instrument that allows Amazon Web Services
                principals to use KMS keys.
                It also can allow them to view a KMS key (DescribeKey) and create and
                manage grants.
                When authorizing access to a KMS key, grants are considered along with
                key policies and IAM policies.
                """);

        waitForInputToContinue(scanner);
        String grantId = grantKey(kmsClient, targetKeyId, granteePrincipal);

```

```
System.out.println("The code granted principal with ARN [" +
granteePrincipal + "] ");
System.out.println("use of the symmetric key. The grant ID is [" + grantId +
"]");
waitForInputToContinue(scanner);

System.out.println(DASHES);
System.out.println("8. List grants for the KMS key");
waitForInputToContinue(scanner);
displayGrantIds(kmsClient, targetKeyId);
waitForInputToContinue(scanner);

System.out.println(DASHES);
System.out.println("9. Revoke the grant");
waitForInputToContinue(scanner);
revokeKeyGrant(kmsClient, targetKeyId, grantId);
waitForInputToContinue(scanner);

System.out.println(DASHES);
System.out.println("10. Decrypt the data\n");
System.out.println("""
    Lets decrypt the data that was encrypted in an early step.
    The code uses the same key to decrypt the string that we encrypted
earlier in the program.
    """);
waitForInputToContinue(scanner);
String decryptText = decryptData(kmsClient, ciphertext, targetKeyId);
System.out.println("Decrypted text is: " + decryptText);
waitForInputToContinue(scanner);

System.out.println(DASHES);
System.out.println("11. Replace a key policy\n");
System.out.println("""
    A key policy is a resource policy for a KMS key. Key policies are the
primary way to control
    access to KMS keys. Every KMS key must have exactly one key policy. The
statements in the key policy
    determine who has permission to use the KMS key and how they can use
it.
    You can also use IAM policies and grants to control access to the KMS
key, but every KMS key
    must have a key policy.
```

By default, when you create a key by using the SDK, a policy is created that gives the AWS account that owns the KMS key full access to the KMS key.

Let's try to replace the automatically created policy with the following policy.

```

        "Version": "2012-10-17",
        "Statement": [{
            "Effect": "Allow",
            "Principal": {"AWS": "arn:aws:iam::000000000000:root"},
            "Action": "kms:*",
            "Resource": "*"
        }]
    """);

    waitForInputToContinue(scanner);
    boolean polAdded = replacePolicy(kmsClient, targetKeyId, policyName);
    waitForInputToContinue(scanner);

    System.out.println(DASHES);
    System.out.println("12. Get the key policy\n");
    System.out.println("The next bit of code that runs gets the key policy to
make sure it exists.");
    waitForInputToContinue(scanner);
    getKeyPolicy(kmsClient, targetKeyId, policyName);
    waitForInputToContinue(scanner);

    System.out.println(DASHES);
    System.out.println("13. Create an asymmetric KMS key and sign your data\n");
    System.out.println("""
        Signing your data with an AWS key can provide several benefits that make
it an attractive option
        for your data signing needs. By using an AWS KMS key, you can leverage
the
        security controls and compliance features provided by AWS,
        which can help you meet various regulatory requirements and enhance the
overall security posture
        of your organization.
    """);
    waitForInputToContinue(scanner);
    signVerifyData(kmsClient);
    waitForInputToContinue(scanner);

```

```
System.out.println(DASHES);
System.out.println("14. Tag your symmetric KMS Key\n");
System.out.println("""
    By using tags, you can improve the overall management, security, and
governance of your
    KMS keys, making it easier to organize, track, and control access to
your encrypted data within
    your AWS environment
    """);
waitForInputToContinue(scanner);
tagKMSKey(kmsClient, targetKeyId);
waitForInputToContinue(scanner);

System.out.println(DASHES);
System.out.println("15. Schedule the deletion of the KMS key\n");
System.out.println("""
    By default, KMS applies a waiting period of 30 days,
    but you can specify a waiting period of 7-30 days. When this operation
is successful,
    the key state of the KMS key changes to PendingDeletion and the key
can't be used in any
    cryptographic operations. It remains in this state for the duration of
the waiting period.

    Deleting a KMS key is a destructive and potentially dangerous operation.
When a KMS key is deleted,
    all data that was encrypted under the KMS key is unrecoverable.\s
    """);
System.out.println("Would you like to delete the Key Management resources?
(y/n)");
String delAns = scanner.nextLine().trim();
if (delAns.equalsIgnoreCase("y")) {
    System.out.println("You selected to delete the AWS KMS resources.");
    waitForInputToContinue(scanner);
    deleteSpecificAlias(kmsClient, fullAliasName);
    disableKey(kmsClient, targetKeyId);
    deleteKey(kmsClient, targetKeyId);
} else {
    System.out.println("The Key Management resources will not be deleted");
}

System.out.println(DASHES);
```

```
        System.out.println("This concludes the AWS Key Management SDK Getting
Started scenario");
        System.out.println(DASHES);
    }
    public static void listAllAliases(KmsClient kmsClient) {
        try {
            ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
                .limit(15)
                .build();

            ListAliasesIterable aliasesResponse =
kmsClient.listAliasesPaginator(aliasesRequest);
            aliasesResponse.stream()
                .flatMap(r -> r.aliases().stream())
                .forEach(alias -> System.out
                    .println("The alias name is: " + alias.aliasName()));

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void disableKey(KmsClient kmsClient, String keyId) {
        try {
            DisableKeyRequest keyRequest = DisableKeyRequest.builder()
                .keyId(keyId)
                .build();

            kmsClient.disableKey(keyRequest);

        } catch (KmsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void signVerifyData(KmsClient kmsClient) {
        String signMessage = "Here is the message that will be digitally signed";

        // Create an AWS KMS key used to digitally sign data.
        CreateKeyRequest request = CreateKeyRequest.builder()
            .keySpec(KeySpec.RSA_2048) // Specify key spec
            .keyUsage(KeyUsageType.SIGN_VERIFY) // Specify key usage
```

```
        .origin(OriginType.AWS_KMS) // Specify key origin
        .build();

CreateKeyResponse response = kmsClient.createKey(request);
String keyId2 = response.keyMetadata().keyId();
System.out.println("Created KMS key with ID: " + keyId2);

SdkBytes bytes = SdkBytes.fromString(signMessage, Charset.defaultCharset());
SignRequest signRequest = SignRequest.builder()
    .keyId(keyId2)
    .message(bytes)
    .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
    .build();

SignResponse signResponse = kmsClient.sign(signRequest);
byte[] signedBytes = signResponse.signature().asByteArray();

// Verify the digital signature.
VerifyRequest verifyRequest = VerifyRequest.builder()
    .keyId(keyId2)

    .message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset())))
    .signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))
    .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
    .build();

VerifyResponse verifyResponse = kmsClient.verify(verifyRequest);
System.out.println("Signature verification result: " +
verifyResponse.signatureValid());
}

public static void tagKMSKey(KmsClient kmsClient, String keyId) {
    try {
        Tag tag = Tag.builder()
            .tagKey("Environment")
            .tagValue("Production")
            .build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .keyId(keyId)
            .tags(tag)
            .build();

        kmsClient.tagResource(tagResourceRequest);
    }
}
```



```
        System.out.println("The key has been tagged.");

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void getKeyPolicy(KmsClient kmsClient, String keyId, String
policyName) {
    try {
        GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
            .keyId(keyId)
            .policyName(policyName)
            .build();

        GetKeyPolicyResponse response = kmsClient.getKeyPolicy(policyRequest);
        System.out.println("The response is "+response.policy());
    } catch (KmsException e) {
        if (e.getMessage().contains("No such policy exists")) {
            System.out.println("The policy cannot be found. Error message: " +
e.getMessage());
        } else {
            throw e;
        }
    }
}

public static boolean replacePolicy(KmsClient kmsClient, String keyId, String
policyName) {
    // Change the principle in the below JSON.
    String policy = ""
    {
        "Version": "2012-10-17",
        "Statement": [{
            "Effect": "Allow",
            "Principal": {"AWS": "arn:aws:iam::%s:root"},
            "Action": "kms:*",
            "Resource": "*"
        }]
    }
    """.formatted(accountId);

    try {
```

```
        PutKeyPolicyRequest keyPolicyRequest = PutKeyPolicyRequest.builder()
            .keyId(keyId)
            .policyName(policyName)
            .policy(policy)
            .build();
        kmsClient.putKeyPolicy(keyPolicyRequest);
        System.out.println("The key policy has been replaced.");
    } catch (LimitExceededException e) {
        System.out.println("Policy limit reached. Unable to create the
policy.");
        return false;
    } catch (AlreadyExistsException e) {
        System.out.println("Only one policy per key is supported. Unable to
create the policy.");
        return false;
    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }

    return true;
}

public static boolean doesKeyHavePolicy(KmsClient kmsClient, String keyId,
String policyName){
    ListKeyPoliciesRequest policiesRequest = ListKeyPoliciesRequest.builder()
        .keyId(keyId)
        .build();

    boolean hasPolicy = false;
    ListKeyPoliciesResponse response =
kmsClient.listKeyPolicies(policiesRequest);
    List<String>policyNames = response.policyNames();
    for (String pol : policyNames) {
        hasPolicy = true;
    }
    return hasPolicy;
}

public static void deleteKey(KmsClient kmsClient, String keyId) {
    try {
        ScheduleKeyDeletionRequest deletionRequest =
ScheduleKeyDeletionRequest.builder()
            .keyId(keyId)
```

```
        .pendingWindowInDays(7)
        .build();

        kmsClient.scheduleKeyDeletion(deletionRequest);
        System.out.println("The key will be deleted in 7 days.");

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void deleteSpecificAlias(KmsClient kmsClient, String aliasName) {
    try {
        DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
            .aliasName(aliasName)
            .build();

        kmsClient.deleteAlias(deleteAliasRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static boolean isKeyEnabled(KmsClient kmsClient, String keyId) {
    try {
        DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
            .keyId(keyId)
            .build();

        DescribeKeyResponse response = kmsClient.describeKey(keyRequest);
        KeyState keyState = response.keyMetadata().keyState();
        if (keyState == KeyState.ENABLED) {
            System.out.println("The key is enabled.");
            return true;
        } else {
            System.out.println("The key is not enabled. Key state: " +
keyState);
        }

    } catch (KmsException e) {
        System.err.println(e.getMessage());
    }
}
```

```
        System.exit(1);
    }
    return false;
}

public static String decryptData(KmsClient kmsClient, SdkBytes encryptedData,
String keyId) {
    try {
        DecryptRequest decryptRequest = DecryptRequest.builder()
            .ciphertextBlob(encryptedData)
            .keyId(keyId)
            .build();

        DecryptResponse decryptResponse = kmsClient.decrypt(decryptRequest);
        return decryptResponse.plaintext().asString(StandardCharsets.UTF_8);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

public static void revokeKeyGrant(KmsClient kmsClient, String keyId, String
grantId) {
    try {
        RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
            .keyId(keyId)
            .grantId(grantId)
            .build();

        kmsClient.revokeGrant(grantRequest);
        System.out.println("Grant ID: [" + grantId + "] was successfully
revoke!");

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void displayGrantIds(KmsClient kmsClient, String keyId) {
    try {
        ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
```

```
        .keyId(keyId)
        .limit(15)
        .build();

    ListGrantsIterable response =
kmsClient.listGrantsPaginator(grantsRequest);
    response.stream()
        .flatMap(r -> r.grants().stream())
        .forEach(grant -> {
            System.out.println("The grant Id is : " + grant.grantId());
            List<GrantOperation> ops = grant.operations();
            for (GrantOperation op : ops) {
                System.out.println(op.name());
            }
        });

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static String grantKey(KmsClient kmsClient, String keyId, String
granteePrincipal) {
    try {
        // Add the desired KMS Grant permissions.
        List<GrantOperation> grantPermissions = new ArrayList<>();
        grantPermissions.add(GrantOperation.ENCRYPT);
        grantPermissions.add(GrantOperation.DECRYPT);
        grantPermissions.add(GrantOperation.DESCRIBE_KEY);

        CreateGrantRequest grantRequest = CreateGrantRequest.builder()
            .keyId(keyId)
            .name("grant1")
            .granteePrincipal(granteePrincipal)
            .operations(grantPermissions)
            .build();

        CreateGrantResponse response = kmsClient.createGrant(grantRequest);
        return response.grantId();

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
    }
    return "";
}

public static void enableKeyRotation(KmsClient kmsClient, String keyId) {
    try {
        EnableKeyRotationRequest enableKeyRotationRequest =
        EnableKeyRotationRequest.builder()
            .keyId(keyId)
            .build();

        kmsClient.enableKeyRotation(enableKeyRotationRequest);
        System.out.println("Key rotation has been enabled for key with id [" +
        keyId + "]");

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void createCustomAlias(KmsClient kmsClient, String targetKeyId,
String aliasName) {
    try {
        CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
            .aliasName(aliasName)
            .targetKeyId(targetKeyId)
            .build();

        kmsClient.createAlias(aliasRequest);
        System.out.println(aliasName + " was successfully created.");

    } catch (ResourceExistsException e) {
        System.err.println("Alias already exists: " + e.getMessage());
        System.err.println("Moving on...");
    } catch (Exception e) {
        System.err.println("An unexpected error occurred: " + e.getMessage());
        System.err.println("Moving on...");
    }
}

public static SdkBytes encryptData(KmsClient kmsClient, String keyId, String
text) {
    try {
```

```
    SdkBytes myBytes = SdkBytes.fromUtf8String(text);
    EncryptRequest encryptRequest = EncryptRequest.builder()
        .keyId(keyId)
        .plaintext(myBytes)
        .build();

    EncryptResponse response = kmsClient.encrypt(encryptRequest);
    String algorithm = response.encryptionAlgorithm().toString();
    System.out.println("The string was encrypted with algorithm " +
algorithm + ".");

    // Get the encrypted data.
    SdkBytes encryptedData = response.ciphertextBlob();
    return encryptedData;

} catch (KmsException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
return null;
}

public static String createKey(KmsClient kmsClient, String keyDesc) {
    try {
        CreateKeyRequest keyRequest = CreateKeyRequest.builder()
            .description(keyDesc)
            .customerMasterKeySpec(CustomerMasterKeySpec.SYMMETRIC_DEFAULT)
            .keyUsage("ENCRYPT_DECRYPT")
            .build();

        CreateKeyResponse result = kmsClient.createKey(keyRequest);
        System.out.println("Symmetric key with ARN [" +
result.keyMetadata().arn() + "] has been created.");
        return result.keyMetadata().keyId();

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

// Enable the KMS key.
public static void enableKey(KmsClient kmsClient, String keyId) {
```

```
    try {
        EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
            .keyId(keyId)
            .build();

        kmsClient.enableKey(enableKeyRequest);

    } catch (KmsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

private static void waitForInputToContinue(Scanner scanner) {
    while (true) {
        System.out.println("");
        System.out.println("Enter 'c' followed by <ENTER> to continue:");
        String input = scanner.nextLine();

        if (input.trim().equalsIgnoreCase("c")) {
            System.out.println("Continuing with the program...");
            System.out.println("");
            break;
        } else {
            // Handle invalid input.
            System.out.println("Invalid input. Please try again.");
        }
    }
}

private static String getAccountId(){
    try (StsClient stsClient = StsClient.create()){
        GetCallerIdentityResponse callerIdentity =
stsClient.getCallerIdentity();
        return callerIdentity.account();
    }
}
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [CreateKey](#)
 - [DescribeKey](#)
 - [DisableKey](#)

- [EnableKey](#)
- [GenerateDataKey](#)
- [ListKeys](#)
- [ScheduleKeyDeletion](#)
- [Sign](#)

用SDK于 Java 2.x 的 Lambda 示例

以下代码示例向您展示了如何使用 with Lambda 来执行操作和实现常见场景。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

开始使用

开始使用 Lambda

以下代码示例展示了如何开始使用 Lambda。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
package com.example.lambda;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.services.lambda.model.LambdaException;
import software.amazon.awssdk.services.lambda.model.ListFunctionsResponse;
import software.amazon.awssdk.services.lambda.model.FunctionConfiguration;
```

```
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListLambdaFunctions {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        LambdaClient awsLambda = LambdaClient.builder()
            .region(region)
            .build();

        listFunctions(awsLambda);
        awsLambda.close();
    }

    public static void listFunctions(LambdaClient awsLambda) {
        try {
            ListFunctionsResponse functionResult = awsLambda.listFunctions();
            List<FunctionConfiguration> list = functionResult.functions();
            for (FunctionConfiguration config : list) {
                System.out.println("The function name is " + config.functionName());
            }
        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListFunctions](#)”中的。

主题

- [操作](#)
- [场景](#)

- [无服务器示例](#)

操作

CreateFunction

以下代码示例演示如何使用 CreateFunction。

SDK适用于 Java 2.x

Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.services.lambda.model.CreateFunctionRequest;
import software.amazon.awssdk.services.lambda.model.FunctionCode;
import software.amazon.awssdk.services.lambda.model.CreateFunctionResponse;
import software.amazon.awssdk.services.lambda.model.GetFunctionRequest;
import software.amazon.awssdk.services.lambda.model.GetFunctionResponse;
import software.amazon.awssdk.services.lambda.model.LambdaException;
import software.amazon.awssdk.services.lambda.model.Runtime;
import software.amazon.awssdk.services.lambda.waiters.LambdaWaiter;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;

/**
 * This code example requires a ZIP or JAR that represents the code of the
 * Lambda function.
 * If you do not have a ZIP or JAR, please refer to the following document:
 *
 * https://github.com/aws-doc-sdk-examples/tree/master/javav2/usecases/
creating_workflows_stepfunctions
 *
 * Also, set up your development environment, including your credentials.
 *
 */
```

```
* For information, see this documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/

public class CreateFunction {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <functionName> <filePath> <role> <handler>\s

            Where:
                functionName - The name of the Lambda function.\s
                filePath - The path to the ZIP or JAR where the code is located.
\s
                role - The role ARN that has Lambda permissions.\s
                handler - The fully qualified method name (for example,
example.Handler::handleRequest). \s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String functionName = args[0];
        String filePath = args[1];
        String role = args[2];
        String handler = args[3];
        Region region = Region.US_WEST_2;
        LambdaClient awsLambda = LambdaClient.builder()
            .region(region)
            .build();

        createLambdaFunction(awsLambda, functionName, filePath, role, handler);
        awsLambda.close();
    }

    public static void createLambdaFunction(LambdaClient awsLambda,
        String functionName,
        String filePath,
        String role,
```

```
String handler) {

    try {
        LambdaWaiter waiter = awsLambda.waiter();
        InputStream is = new FileInputStream(filePath);
        SdkBytes fileToUpload = SdkBytes.fromInputStream(is);

        FunctionCode code = FunctionCode.builder()
            .zipFile(fileToUpload)
            .build();

        CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
            .functionName(functionName)
            .description("Created by the Lambda Java API")
            .code(code)
            .handler(handler)
            .runtime(Runtime.JAVA8)
            .role(role)
            .build();

        // Create a Lambda function using a waiter.
        CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
        GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
            .functionName(functionName)
            .build();
        WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("The function ARN is " +
functionResponse.functionArn());

    } catch (LambdaException | FileNotFoundException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateFunction](#)”中的。

DeleteFunction

以下代码示例演示如何使用 DeleteFunction。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.model.DeleteFunctionRequest;
import software.amazon.awssdk.services.lambda.model.LambdaException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteFunction {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <functionName>\s

                Where:
                functionName - The name of the Lambda function.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String functionName = args[0];
        Region region = Region.US_EAST_1;
```

```
        LambdaClient awsLambda = LambdaClient.builder()
            .region(region)
            .build();

        deleteLambdaFunction(awsLambda, functionName);
        awsLambda.close();
    }

    public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName) {
        try {
            DeleteFunctionRequest request = DeleteFunctionRequest.builder()
                .functionName(functionName)
                .build();

            awsLambda.deleteFunction(request);
            System.out.println("The " + functionName + " function was deleted");

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteFunction](#)”中的。

Invoke

以下代码示例演示如何使用 Invoke。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import org.json.JSONObject;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
```

```
import software.amazon.awssdk.services.lambda.LambdaClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lambda.model.InvokeRequest;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.lambda.model.InvokeResponse;
import software.amazon.awssdk.services.lambda.model.LambdaException;

public class LambdaInvoke {

    /**
     * Function names appear as
     * arn:aws:lambda:us-west-2:335556666777:function:HelloFunction
     * you can retrieve the value by looking at the function in the AWS Console
     *
     * Also, set up your development environment, including your credentials.
     *
     * For information, see this documentation topic:
     *
     * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
     */

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <functionName>\s

            Where:
                functionName - The name of the Lambda function\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String functionName = args[0];
        Region region = Region.US_WEST_2;
        LambdaClient awsLambda = LambdaClient.builder()
            .region(region)
            .build();

        invokeFunction(awsLambda, functionName);
    }
}
```



```
        awsLambda.close();
    }

    public static void invokeFunction(LambdaClient awsLambda, String functionName) {

        InvokeResponse res = null;
        try {
            // Need a SdkBytes instance for the payload.
            JSONObject jsonObj = new JSONObject();
            jsonObj.put("inputValue", "2000");
            String json = jsonObj.toString();
            SdkBytes payload = SdkBytes.fromUtf8String(json);

            // Setup an InvokeRequest.
            InvokeRequest request = InvokeRequest.builder()
                .functionName(functionName)
                .payload(payload)
                .build();

            res = awsLambda.invoke(request);
            String value = res.payload().asUtf8String();
            System.out.println(value);

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参见在AWS SDK for Java 2.x API参考中[调用](#)。

场景

函数入门


以下代码示例展示了如何：

- 创建IAM角色和 Lambda 函数，然后上传处理程序代码。
- 使用单个参数来调用函数并获取结果。
- 更新函数代码并使用环境变量进行配置。

- 使用新参数来调用函数并获取结果。显示返回的执行日志。
- 列出账户函数，然后清除函数。

有关更多信息，请参阅[使用控制台创建 Lambda 函数](#)。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/*
 * Lambda function names appear as:
 *
 * arn:aws:lambda:us-west-2:335556666777:function:HelloFunction
 *
 * To find this value, look at the function in the AWS Management Console.
 *
 * Before running this Java code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example performs the following tasks:
 *
 * 1. Creates an AWS Lambda function.
 * 2. Gets a specific AWS Lambda function.
 * 3. Lists all Lambda functions.
 * 4. Invokes a Lambda function.
 * 5. Updates the Lambda function code and invokes it again.
 * 6. Updates a Lambda function's configuration value.
 * 7. Deletes a Lambda function.
 */

public class LambdaScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
```

```
public static void main(String[] args) throws InterruptedException {
    final String usage = ""

        Usage:
            <functionName> <filePath> <role> <handler> <bucketName> <key>\s

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

    if (args.length != 6) {
        System.out.println(usage);
        System.exit(1);
    }

    String functionName = args[0];
    String filePath = args[1];
    String role = args[2];
    String handler = args[3];
    String bucketName = args[4];
    String key = args[5];

    Region region = Region.US_WEST_2;
    LambdaClient awsLambda = LambdaClient.builder()
        .region(region)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the AWS Lambda example scenario.");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("1. Create an AWS Lambda function.");
```

```
String funArn = createLambdaFunction(awsLambda, functionName, filePath,
role, handler);
System.out.println("The AWS Lambda ARN is " + funArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Get the " + functionName + " AWS Lambda function.");
getFunction(awsLambda, functionName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. List all AWS Lambda functions.");
listFunctions(awsLambda);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Invoke the Lambda function.");
System.out.println("*** Sleep for 1 min to get Lambda function ready.");
Thread.sleep(60000);
invokeFunction(awsLambda, functionName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Update the Lambda function code and invoke it
again.");
updateFunctionCode(awsLambda, functionName, bucketName, key);
System.out.println("*** Sleep for 1 min to get Lambda function ready.");
Thread.sleep(60000);
invokeFunction(awsLambda, functionName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Update a Lambda function's configuration value.");
updateFunctionConfiguration(awsLambda, functionName, handler);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Delete the AWS Lambda function.");
LambdaScenario.deleteLambdaFunction(awsLambda, functionName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The AWS Lambda scenario completed successfully");
System.out.println(DASHES);
```

```
        awsLambda.close();
    }

    public static String createLambdaFunction(LambdaClient awsLambda,
        String functionName,
        String filePath,
        String role,
        String handler) {

        try {
            LambdaWaiter waiter = awsLambda.waiter();
            InputStream is = new FileInputStream(filePath);
            SdkBytes fileToUpload = SdkBytes.fromInputStream(is);

            FunctionCode code = FunctionCode.builder()
                .zipFile(fileToUpload)
                .build();

            CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
                .functionName(functionName)
                .description("Created by the Lambda Java API")
                .code(code)
                .handler(handler)
                .runtime(Runtime.JAVA8)
                .role(role)
                .build();

            // Create a Lambda function using a waiter
            CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
            GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
                .functionName(functionName)
                .build();

            WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            return functionResponse.functionArn();

        } catch (LambdaException | FileNotFoundException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
}
```

```
public static void getFunction(LambdaClient awsLambda, String functionName) {
    try {
        GetFunctionRequest functionRequest = GetFunctionRequest.builder()
            .functionName(functionName)
            .build();

        GetFunctionResponse response = awsLambda.getFunction(functionRequest);
        System.out.println("The runtime of this Lambda function is " +
response.configuration().runtime());

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void listFunctions(LambdaClient awsLambda) {
    try {
        ListFunctionsResponse functionResult = awsLambda.listFunctions();
        List<FunctionConfiguration> list = functionResult.functions();
        for (FunctionConfiguration config : list) {
            System.out.println("The function name is " + config.functionName());
        }

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void invokeFunction(LambdaClient awsLambda, String functionName) {

    InvokeResponse res;
    try {
        // Need a SdkBytes instance for the payload.
        JSONObject jsonObj = new JSONObject();
        jsonObj.put("inputValue", "2000");
        String json = jsonObj.toString();
        SdkBytes payload = SdkBytes.fromUtf8String(json);

        InvokeRequest request = InvokeRequest.builder()
            .functionName(functionName)
            .payload(payload)
    }
```

```
        .build();

        res = awsLambda.invoke(request);
        String value = res.payload().asUtf8String();
        System.out.println(value);

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateFunctionCode(LambdaClient awsLambda, String
functionName, String bucketName, String key) {
    try {
        LambdaWaiter waiter = awsLambda.waiter();
        UpdateFunctionCodeRequest functionCodeRequest =
UpdateFunctionCodeRequest.builder()
            .functionName(functionName)
            .publish(true)
            .s3Bucket(bucketName)
            .s3Key(key)
            .build();

        UpdateFunctionCodeResponse response =
awsLambda.updateFunctionCode(functionCodeRequest);
        GetFunctionConfigurationRequest getFunctionConfigRequest =
GetFunctionConfigurationRequest.builder()
            .functionName(functionName)
            .build();

        WaiterResponse<GetFunctionConfigurationResponse> waiterResponse = waiter
            .waitUntilFunctionUpdated(getFunctionConfigRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("The last modified value is " +
response.lastModified());

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
public static void updateFunctionConfiguration(LambdaClient awsLambda, String
functionName, String handler) {
    try {
        UpdateFunctionConfigurationRequest configurationRequest =
UpdateFunctionConfigurationRequest.builder()
            .functionName(functionName)
            .handler(handler)
            .runtime(Runtime.JAVA11)
            .build();

        awsLambda.updateFunctionConfiguration(configurationRequest);

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName) {
    try {
        DeleteFunctionRequest request = DeleteFunctionRequest.builder()
            .functionName(functionName)
            .build();

        awsLambda.deleteFunction(request);
        System.out.println("The " + functionName + " function was deleted");

    } catch (LambdaException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。

- [CreateFunction](#)
- [DeleteFunction](#)
- [GetFunction](#)
- [Invoke](#)

- [ListFunctions](#)
- [UpdateFunctionCode](#)
- [UpdateFunctionConfiguration](#)

无服务器示例

使用 Lambda 函数连接到亚马逊RDS数据库

以下代码示例说明如何实现连接到数据库的 Lambda 函数。RDS该函数发出一个简单的数据库请求并返回结果。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

使用 Java 在 Lambda 函数中连接到亚马逊RDS数据库。

```
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();
}
}
```

通过 Kinesis 触发器调用 Lambda 函数

以下代码示例展示了如何实现一个 Lambda 函数，该函数接收因接收来自 Kinesis 流的记录而触发的事件。该函数检索 Kinesis 有效负载，将 Base64 解码，并记录下记录内容。

SDK适用于 Java 2.x

Note

还有更多相关信息 [GitHub](#)。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

使用 Java 将 Kinesis 事件与 Lambda 结合使用。

```
// 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;
    }
}
```

通过 DynamoDB 触发器调用 Lambda 函数

以下代码示例演示如何实现 Lambda 函数，该函数接收通过从 DynamoDB 流接收记录而触发的事件。该函数检索 DynamoDB 有效负载，并记录下记录内容。

SDK适用于 Java 2.x

Note

还有更多相关信息 [GitHub](#)。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

使用 Java 将 DynamoDB 事件与 Lambda 结合使用。

```
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()));
    }
}
```

通过 Amazon S3 触发器调用 Lambda 函数

以下代码示例展示了如何实现一个 Lambda 函数，该函数接收通过将对象上传到 S3 桶而触发的事件。该函数从事件参数中检索 S3 存储桶名称和对象密钥，并调用 Amazon S3 API 来检索和记录对象的内容类型。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

使用 Java 将 S3 事件与 Lambda 结合使用。

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

从亚马逊触发器调用 Lambda 函数 SNS

以下代码示例说明如何实现一个 Lambda 函数，该函数接收通过接收来自主题的消息而触发的事件。SNS 该函数从事件参数检索消息并记录每条消息的内容。

SDK 适用于 Java 2.x

Note

还有更多相关信息 [GitHub](#)。在 [无服务器示例](#) 存储库中查找完整示例，并了解如何进行设置和运行。

使用 Java 在 Lambda 上使用 SNS 事件。

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

从亚马逊触发器调用 Lambda 函数 SQS

以下代码示例说明如何实现一个 Lambda 函数，该函数接收通过从队列接收消息而触发的事件。SQS 该函数从事件参数检索消息并记录每条消息的内容。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

使用 Java 在 Lambda 上使用 SQS 事件。

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

通过 Kinesis 触发器报告 Lambda 函数批处理项目失败

以下代码示例展示了如何为接收来自 Kinesis 流的事件的 Lambda 函数实现部分批处理响应。该函数在响应中报告批处理项目失败，并指示 Lambda 稍后重试这些消息。

SDK适用于 Java 2.x

Note

还有更多相关信息 [GitHub](#)。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

报告使用 Java 进行 Lambda Kinesis 批处理项目失败。

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

通过 DynamoDB 触发器报告 Lambda 函数批处理项目失败

以下代码示例演示如何为接收来自 DynamoDB 流的事件的 Lambda 函数实现部分批量响应。该函数在响应中报告批处理项目失败，并指示 Lambda 稍后重试这些消息。

SDK适用于 Java 2.x

Note

还有更多相关信息 [GitHub](#)。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

报告使用 Java 通过 Lambda 进行 DynamoDB 批处理项目失败。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.DynamodbEvent;
import com.amazonaws.services.lambda.runtime.events.StreamsEventResponse;
import com.amazonaws.services.lambda.runtime.events.models.dynamodb.StreamRecord;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;

public class ProcessDynamodbRecords implements RequestHandler<DynamodbEvent,
    Serializable> {

    @Override
    public StreamsEventResponse handleRequest(DynamodbEvent input, Context context)
    {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
        ArrayList<>();
        String curRecordSequenceNumber = "";

        for (DynamodbEvent.DynamodbStreamRecord dynamodbStreamRecord :
        input.getRecords()) {
```

```
try {
    //Process your record
    StreamRecord dynamodbRecord = dynamodbStreamRecord.getDynamodb();
    curRecordSequenceNumber = dynamodbRecord.getSequenceNumber();

} catch (Exception e) {
    /* Since we are working with streams, we can return the failed item
    immediately.
        Lambda will immediately begin to retry processing from this
    failed item onwards. */
    batchItemFailures.add(new
StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
    return new StreamsEventResponse(batchItemFailures);
}
}

return new StreamsEventResponse();
}
```

使用亚马逊触发器报告 Lambda 函数的批量项目失败 SQS

以下代码示例说明如何为从队列接收事件的 Lambda 函数实现部分批量响应。SQS 该函数在响应中报告批处理项目失败，并指示 Lambda 稍后重试这些消息。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

使用 Java 使用 Lambda 报告SQS批处理项目失败。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
```

```
import com.amazonaws.services.lambda.runtime.events.SQSBatchResponse;

import java.util.ArrayList;
import java.util.List;

public class ProcessSQSMessageBatch implements RequestHandler<SQSEvent,
SQSBatchResponse> {
    @Override
    public SQSBatchResponse handleRequest(SQSEvent sqsEvent, Context context) {

        List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
ArrayList<SQSBatchResponse.BatchItemFailure>();
        String messageId = "";
        for (SQSEvent.SQSMessage message : sqsEvent.getRecords()) {
            try {
                //process your message
                messageId = message.getMessageId();
            } catch (Exception e) {
                //Add failed message identifier to the batchItemFailures list
                batchItemFailures.add(new
SQSBatchResponse.BatchItemFailure(messageId));
            }
        }
        return new SQSBatchResponse(batchItemFailures);
    }
}
```

AWS Marketplace SDK适用于 Java 2.x 的协议服务示例

以下代码示例向您展示了如何使用 with Agreement Service 来执行操作和实现常见场景。AWS SDK for Java 2.x AWS Marketplace

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [协议](#)

协议

获得所有同意 IDs

以下代码示例显示了如何获得所有同意IDs。

SDK适用于 Java 2.x

```
// 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;
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [SearchAgreements](#)”中的。

获取所有协议

以下代码示例显示了如何获取所有协议。

SDK适用于 Java 2.x

```
// 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;
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [SearchAgreements](#)”中的。

从协议中获取客户 ID

以下代码示例显示如何从协议中获取客户 ID。

SDK适用于 Java 2.x

```
// 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;
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeAgreement](#)”中的。

从协议中获取财务细节

以下代码示例显示如何从协议中获取财务细节。

SDK适用于 Java 2.x

```
// 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;
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeAgreement](#)”中的。

从协议中获取免费试用详情

以下代码示例显示如何从协议中获取免费试用详细信息。

SDK适用于 Java 2.x

```
// 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;
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeAgreement](#)”中的。

获取有关协议的信息

以下代码示例显示了如何获取有关协议的信息。

SDK适用于 Java 2.x

```
// 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;
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeAgreement](#)”中的。

从协议中获取产品和优惠详情

以下代码示例说明如何从协议中获取产品和报价详情。

SDK适用于 Java 2.x

```
// 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;
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeAgreement](#)”中的。

EULA达成协议

以下代码示例显示了如何获取协议。EULA

SDK适用于 Java 2.x

```
// 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;
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetAgreementTerms](#)”中的。

获取协议的 auto 续订条款

以下代码示例显示了如何获取协议的 auto 续订条款。

SDK适用于 Java 2.x

```
// 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;
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetAgreementTerms](#)”中的。

获取在协议中购买的尺寸

以下代码示例显示了如何获取在协议中购买的维度。

SDK适用于 Java 2.x

```
// 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;
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetAgreementTerms](#)”中的。

获取在协议中购买的每个维度的实例

以下代码示例显示了如何获取在协议中购买的每个维度的实例。

SDK适用于 Java 2.x

```
// 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;
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetAgreementTerms](#)”中的。

获取协议的付款时间表

以下代码示例显示了如何获取协议的付款时间表。

SDK适用于 Java 2.x

```
// 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;
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetAgreementTerms](#)”中的。

在协议中获取每个维度的定价

以下代码示例显示了如何在协议中获取每个维度的定价。

SDK适用于 Java 2.x

```
// 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;
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetAgreementTerms](#)”中的。

获取协议的定价类型

以下代码示例显示了如何获取协议的定价类型。

SDK适用于 Java 2.x

```
// 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).bui

```

```
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;
}
}
```


- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeAgreement](#)”中的。

获取协议的产品类型

以下代码示例显示了如何获取协议的产品类型。

SDK适用于 Java 2.x

```
// 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;
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeAgreement](#)”中的。

获取协议的状态

以下代码示例显示了如何获取协议的状态。

SDK适用于 Java 2.x

```
// 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;
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeAgreement](#)”中的。

获取协议的支持条款

以下代码示例显示了如何获取协议的支持条款。

SDK适用于 Java 2.x

```
// 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;
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetAgreementTerms](#)”中的。

获取协议条款

以下代码示例显示了如何获取协议条款。

SDK适用于 Java 2.x

```
// 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;
    }

}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetAgreementTerms](#)”中的。

按结束日期搜索协议

以下代码示例说明如何按结束日期搜索协议。

SDK适用于 Java 2.x

```
// 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;
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [SearchAgreements](#)”中的。

使用一个自定义筛选器搜索协议

以下代码示例演示如何使用一个自定义筛选器搜索协议。

SDK适用于 Java 2.x

```
// 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;
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [SearchAgreements](#)”中的。

使用两个自定义筛选器搜索协议

以下代码示例说明如何使用两个自定义筛选器搜索协议。

SDK适用于 Java 2.x

```
// 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;
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [SearchAgreements](#)”中的。

AWS Marketplace Catalog API 用SDK于 Java 2.x 的示例

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS Marketplace Catalog API。

AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [AMI产品](#)
- [渠道合作伙伴优惠](#)
- [容器产品](#)
- [实体](#)
- [Offers](#)
- [产品](#)
- [转售授权](#)
- [SaaS 产品](#)
- [实用程序](#)

AMI产品

为现有AMI产品添加维度并更新优惠定价条款

以下代码示例显示了如何为现有AMI产品添加维度并更新优惠定价条款。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```
{
```

```

"Catalog": "AWSMarketplace",
"ChangeSet": [
  {
    "ChangeType": "AddDimensions",
    "Entity": {
      "Identifier": "prod-111111111111",
      "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-111111111111"
    },
    "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"
  }
]
}
]
}
]
}
]
}
]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

添加部署AMI产品的区域

以下代码示例显示了如何添加部署AMI产品的区域。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "AddRegions",
      "Entity": {
        "Type": "AmiProduct@1.0",
        "Identifier": "prod-111111111111"
      },
      "DetailsDocument": {
        "Regions": [
          "us-east-2",
          "us-west-2"
        ]
      }
    }
  ]
}

```

```

    }
  ]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

创建按小时按年定价的公开或限量AMI产品以及公开发售

以下代码示例显示了如何创建按小时按年定价的公开AMI产品或限量产品以及公开发售。此示例创建标准或自定义EULA。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "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/awsmvp-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-111111111111111111",
                        "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": {}
}
]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

创建按小时按月定价的公开或限量AMI产品和公开报价

以下代码示例展示了如何创建按小时按月定价的公开或限量AMI产品以及公开报价。此示例创建标准或自定义EULA。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "AmiProduct@1.0",

```



```

        "Identifier": "$CreateProductChange.Entity.Identifier"
    },
    "DetailsDocument": {
        "ProductTitle": "Sample product",
        "ShortDescription": "Brief description",
        "LongDescription": "Detailed description",
        "Highlights": [
            "Sample highlight"
        ],
        "SearchKeywords": [
            "Sample keyword"
        ],
        "Categories": [
            "Operating Systems"
        ],
        "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
        "VideoUrls": [
            "https://sample.amazonaws.com/awsmvp-video-1"
        ],
        "AdditionalResources": []
    }
},
{
    "ChangeType": "AddRegions",
    "Entity": {
        "Type": "AmiProduct@1.0",
        "Identifier": "$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-111111111111111111",
                "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": {}
    }
]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

创建按小时定价的公开或限量AMI产品和公开报价

以下代码示例展示了如何创建按小时定价的公开或限量AMI产品以及公开发售。此示例创建了 `and` 标准或自定义EULA。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "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/awsmvp-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": "$CreateProductChange.Entity.Identifier"
    },
    "DetailsDocument": {
        "Version": {
            "VersionTitle": "Test AMI Version1.0",
            "ReleaseNotes": "Test AMI Version"
        },
        "DeliveryOptions": [
            {
                "Details": {
                    "AmiDeliveryOptionDetails": {
                        "AmiSource": {
                            "AmiId": "ami-111111111111111111",
                            "AccessRoleArn":
"arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                            "UserName": "ec2-user",
                            "OperatingSystemName": "AMAZONLINUX",
                            "OperatingSystemVersion": "10.0.14393",
                            "ScanningPort": 22
                        },
                        "UsageInstructions": "Test AMI Version",
                        "RecommendedInstanceType": "t2.micro",
                        "SecurityGroups": [
                            {
                                "IpProtocol": "tcp",
                                "IpRanges": [
                                    "0.0.0.0/0"
                                ],
                                "FromPort": 10,
                                "ToPort": 22
                            }
                        ]
                    }
                }
            }
        ]
    },
    "ChangeType": "AddDimensions",
    "Entity": {
        "Type": "AmiProduct@1.0",
        "Identifier": "$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": {}
}
]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

使用公开报价草稿创建AMI产品草稿

以下代码示例展示了如何使用公开报价草稿创建AMI产品草稿。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```
{
  "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"
      }
    }
  ]
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

限制部署AMI产品的区域

以下代码示例显示了如何限制部署AMI产品的区域。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```
{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "RestrictRegions",
      "Entity": {
        "Type": "AmiProduct@1.0",
        "Identifier": "prod-111111111111"
      },
      "DetailsDocument": {
        "Regions": [
          "us-west-2"
        ]
      }
    }
  ]
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

限制产品知名度

以下代码示例显示了如何限制产品的可见性。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```
{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "UpdateVisibility",
      "Entity": {
        "Type": "AmiProduct@1.0",
```

```

        "Identifier": "prod-111111111111"
    },
    "DetailsDocument": {
        "TargetVisibility": "Restricted"
    }
}
]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

指定是否在新区域部署AMI资产

以下代码示例显示了如何指定是否在为支持 future 区域而 AWS 构建的新区域中部署AMI资产。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "UpdateFutureRegionSupport",
      "Entity": {
        "Type": "AmiProduct@1.0",
        "Identifier": "prod-111111111111"
      },
      "DetailsDocument": {
        "FutureRegionSupport": {
          "SupportedRegions": [
            "All"
          ]
        }
      }
    }
  ]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

渠道合作伙伴优惠

CPPO为任何产品类型创建草稿

以下代码示例展示了如何CPPO为任何产品类型创建草稿，这样您就可以在向买家发布产品之前在内部对其进行审核。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```
{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "CreateOfferUsingResaleAuthorization",
      "Entity": {
        "Type": "Offer@1.0"
      },
      "DetailsDocument": {
        "ResaleAuthorizationId": "11111111-1111-1111-1111-111111111111",
        "Name": "Test Offer",
        "Description": "Test product"
      }
    }
  ]
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

使用合同定价创建转售授权替代私募报价

以下代码示例显示了如何根据合同定价的现有协议创建转售授权替代私募报价。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```
{
  "Catalog": "AWSMarketplace",
```

```

"ChangeSet": [
  {
    "ChangeType" : "CreateReplacementOfferUsingResaleAuthorization",
    "Entity": {
      "Type": "Offer@1.0"
    },
    "ChangeName": "CreateReplacementOfferResaleAuth",
    "DetailsDocument": {
      "AgreementId": "agmt-11111111111111111111111111111111",
      "ResaleAuthorizationId": "resaleauthz-1111111111111111"
    }
  },
  {
    "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": {}
}
]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

列出渠道合作伙伴CPPOs创建的所有内容

以下代码示例展示了如何列出渠道合作伙伴CPPOs创建的所有内容。

SDK适用于 Java 2.x

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
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 CPP0 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;
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListEntities](#)”中的。

列出渠道合作伙伴可用的所有共享转售授权

以下代码示例显示如何列出渠道合作伙伴可用的所有共享转售授权。

SDK适用于 Java 2.x

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
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;
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListEntities](#)”中的。

发布CPPO并追加买家 EULA

以下代码示例展示了如何发布CPPO和追加买家EULA。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```
{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "CreateOfferUsingResaleAuthorization",
      "Entity": {
        "Type": "Offer@1.0"
      },
      "ChangeName": "CreateCPP0offer",
      "DetailsDocument": {
        "ResaleAuthorizationId": "resaleauthz-111111111111",
        "Name": "Test Offer",
        "Description": "Test product"
      }
    },
    {
      "ChangeType": "UpdateLegalTerms",
      "Entity": {
        "Type": "Offer@1.0",
        "Identifier": "$CreateCPP0offer.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": "$CreateCPP0offer.Entity.Identifier"
  },
  "DetailsDocument": {
    "PositiveTargeting": {
      "BuyerAccounts": ["222222222222"]
    }
  }
},
{
  "ChangeType": "UpdateAvailability",
  "Entity": {
    "Type": "Offer@1.0",
    "Identifier": "$CreateCPP0offer.Entity.Identifier"
  },
  "DetailsDocument": {
    "AvailabilityEndDate": "2023-07-31"
  }
},
{
  "ChangeType": "UpdateValidityTerms",
  "Entity": {
    "Type": "Offer@1.0",
    "Identifier": "$CreateCPP0offer.Entity.Identifier"
  },
  "DetailsDocument": {
    "Terms": [

```



```

        {
            "Type": "ValidityTerm",
            "AgreementDuration": "P450D"
        }
    ]
},
{
    "ChangeType": "ReleaseOffer",
    "Entity": {
        "Type": "Offer@1.0",
        "Identifier": "$CreateCPP0offer.Entity.Identifier"
    },
    "DetailsDocument": {}
}
]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

发布CPPO使用一次性转售授权并更新价格加价

以下代码示例展示了如何在 SaaS 或容器产品上AMI发布一次性转售授权以及如何更新价格加价。CPPO

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOfferUsingResaleAuthorization",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPP0offer",
            "DetailsDocument": {
                "ResaleAuthorizationId": "resaleauthz-111111111111",
                "Name": "Test Offer",
                "Description": "Test product"
            }
        }
    ]
}

```

```
    }
  },
  {
    "ChangeType": "UpdateMarkup",
    "Entity": {
      "Type": "Offer@1.0",
      "Identifier": "$CreateCPP0offer.Entity.Identifier"
    },
    "DetailsDocument": {
      "Percentage" : "5.0"
    }
  },
  {
    "ChangeType": "UpdateTargeting",
    "Entity": {
      "Type": "Offer@1.0",
      "Identifier": "$CreateCPP0offer.Entity.Identifier"
    },
    "DetailsDocument": {
      "PositiveTargeting": {
        "BuyerAccounts": ["222222222222"]
      }
    }
  },
  {
    "ChangeType": "UpdateAvailability",
    "Entity": {
      "Type": "Offer@1.0",
      "Identifier": "$CreateCPP0offer.Entity.Identifier"
    },
    "DetailsDocument": {
      "AvailabilityEndDate": "2023-07-31"
    }
  },
  {
    "ChangeType": "UpdateValidityTerms",
    "Entity": {
      "Type": "Offer@1.0",
      "Identifier": "$CreateCPP0offer.Entity.Identifier"
    },
    "DetailsDocument": {
      "Terms": [
        {
          "Type": "ValidityTerm",
```

```

        "AgreementDuration": "P450D"
    }
    ]
}
},
{
    "ChangeType": "ReleaseOffer",
    "Entity": {
        "Type": "Offer@1.0",
        "Identifier": "$CreateCPP0offer.Entity.Identifier"
    },
    "DetailsDocument": {}
}
]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

发布草稿CPPO并更新价格加价

以下代码示例展示了如何发布草稿CPPO和更新价格加价。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOfferUsingResaleAuthorization",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPP0offer",
            "DetailsDocument": {
                "ResaleAuthorizationId": "resaleauthz-111111111111",
                "Name": "Test Offer",
                "Description": "Test product"
            }
        },
        {

```

```

    "ChangeType": "UpdateMarkup",
    "Entity": {
      "Type": "Offer@1.0",
      "Identifier": "$CreateCPP0offer.Entity.Identifier"
    },
    "DetailsDocument": {
      "Percentage" : "5.0"
    }
  },
  {
    "ChangeType": "UpdateTargeting",
    "Entity": {
      "Type": "Offer@1.0",
      "Identifier": "$CreateCPP0offer.Entity.Identifier"
    },
    "DetailsDocument": {
      "PositiveTargeting": {
        "BuyerAccounts": ["222222222222"]
      }
    }
  },
  {
    "ChangeType": "UpdateAvailability",
    "Entity": {
      "Type": "Offer@1.0",
      "Identifier": "$CreateCPP0offer.Entity.Identifier"
    },
    "DetailsDocument": {
      "AvailabilityEndDate": "2023-07-31"
    }
  },
  {
    "ChangeType": "UpdateValidityTerms",
    "Entity": {
      "Type": "Offer@1.0",
      "Identifier": "$CreateCPP0offer.Entity.Identifier"
    },
    "DetailsDocument": {
      "Terms": [
        {
          "Type": "ValidityTerm",
          "AgreementDuration": "P450D"
        }
      ]
    }
  }
]

```

```

    }
  },
  {
    "ChangeType": "ReleaseOffer",
    "Entity": {
      "Type": "Offer@1.0",
      "Identifier": "$CreateCPP0offer.Entity.Identifier"
    },
    "DetailsDocument": {}
  }
]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

更新 a 的到期日期 CPPO

以下代码示例说明如何更新到期日期，让买家有CPPO更多时间评估和接受报价。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "UpdateAvailability",
      "Entity": {
        "Type": "Offer@1.0",
        "Identifier": "offer-11111111111111"
      },
      "DetailsDocument": {
        "AvailabilityEndDate": "2025-07-31"
      }
    }
  ]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

容器产品

使用公开报价草稿创建集装箱产品草稿

以下代码示例展示了如何使用公开报价草稿创建集装箱产品草稿。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```
{
  "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"
      }
    }
  ]
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

创建具有公开报价和合同定价的限量集装箱产品

以下代码示例展示了如何创建具有公开报价、合同定价和标准的限量容器产品EULA。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```
{
  "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": {}
    }
  ]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

实体

在一次通话中描述所有实体

以下代码示例显示了如何在单个调用中描述所有实体。

SDK适用于 Java 2.x

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

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [BatchDescribeEntities](#)”中的。

列出并描述与商品相关的所有报价

以下代码示例说明如何列出和描述与产品相关的所有报价。

SDK适用于 Java 2.x

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
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;
}
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [DescribeEntity](#)
 - [ListEntities](#)

Offers

为 SaaS 产品创建自定义维度并创建私人报价

以下代码示例展示了如何为 SaaS 产品创建自定义维度并创建私有报价。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```
{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "AddDimensions",
      "Entity": {
        "Type": "SaaSProduct@1.0",
```



```

        "Identifier": "prod-111111111111"
    },
    "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-111111111111"
    },
    "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"
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

为AMI或 SaaS 产品创建私募报价草稿

以下代码示例展示了如何为AMI或 SaaS 产品创建私募报价草稿，这样您就可以在向买家发布之前在内部对其进行审核。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```
{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "CreateOffer",
      "Entity": {
        "Type": "Offer@1.0"
      },
      "DetailsDocument": {
        "ProductId": "prod-111111111111",
        "Name": "Test Private Offer"
      }
    }
  ]
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

为 SaaS 产品创建包含合同和即用即付定价的私人报价

以下代码示例演示如何使用合同和即用即付定价为 SaaS 产品创建私人报价。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```
{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "CreateOffer",
```

```

    "Entity": {
      "Type": "Offer@1.0"
    },
    "ChangeName": "CreateOfferChange",
    "DetailsDocument": {
      "ProductId": "prod-111111111111"
    }
  },
  {
    "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": {}
}
]

```

```
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

为 SaaS 产品创建具有合同定价和灵活付款时间表的私有报价

以下代码示例展示了如何为 SaaS 产品创建具有合同定价和灵活付款计划的私募报价。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```
{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "CreateOffer",
      "Entity": {
        "Type": "Offer@1.0"
      },
      "ChangeName": "CreateOfferChange",
      "DetailsDocument": {
        "ProductId": "prod-11111111111111"
      }
    },
    {
      "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": {}
    }
]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

为容器产品创建带有合同定价的私有报价

以下代码示例展示了如何为容器产品创建具有合同定价的私募报价。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "CreateOffer",
      "Entity": {
        "Type": "Offer@1.0"
      },
      "ChangeName": "CreateOfferChange",
      "DetailsDocument": {
        "ProductId": "prod-11111111111111"
      }
    },
    {
      "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": [
            "11111111111111"
          ]
        }
      }
    },
    {
      "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": {}
}
]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

为AMI产品创建包含合同定价的私有报价

以下代码示例显示了如何为AMI产品创建具有合同定价的私募报价。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {

```

```

    "ChangeType": "CreateOffer",
    "ChangeName": "CreateOfferChange",
    "Entity": {
      "Type": "Offer@1.0"
    },
    "DetailsDocument": {
      "ProductId": "prod-111111111111"
    }
  },
  {
    "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": {}
}
]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

为AMI产品创建具有按小时按年定价和灵活付款时间表的私人报价

以下代码示例展示了如何为AMI产品创建具有按小时按年定价和灵活付款时间表的私人报价。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "CreateOffer",
      "ChangeName": "CreateOfferChange",
      "Entity": {

```

```

        "Type": "Offer@1.0"
    },
    "DetailsDocument": {
        "ProductId": "prod-111111111111"
    }
},
{
    "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": {}
}
]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

为AMI产品创建按小时按年定价的私人报价

以下代码示例展示了如何为AMI产品创建按小时按年定价的私人报价。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "CreateOffer",
      "ChangeName": "CreateOfferChange",
      "Entity": {
        "Type": "Offer@1.0"
      },
      "DetailsDocument": {

```

```

        "ProductId": "prod-111111111111"
    },
    {
        "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": {}
    }
}

```



```

    }
  ]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

为AMI产品创建按小时定价的私人报价

以下代码示例展示了如何为AMI产品创建按小时定价的私有报价。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "CreateOffer",
      "ChangeName": "CreateOfferChange",
      "Entity": {
        "Type": "Offer@1.0"
      },
      "DetailsDocument": {
        "ProductId": "prod-111111111111"
      }
    },
    {
      "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": {}
}
]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

为 SaaS 产品创建具有订阅定价的私有报价

以下代码示例说明如何为 SaaS 产品创建具有订阅定价的私有报价。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "CreateOffer",
      "Entity": {
        "Type": "Offer@1.0"
      },
      "ChangeName": "CreateOfferChange",
      "DetailsDocument": {
        "ProductId": "prod-11111111111111"
      }
    },
    {
      "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": {}
    }
]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

为 SaaS 产品创建具有分级合同定价的私有报价

以下代码示例展示了如何为 SaaS 产品创建具有分级合同定价的私募报价。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "CreateOffer",
      "Entity": {
        "Type": "Offer@1.0"
      },
      "ChangeName": "CreateOfferChange",
      "DetailsDocument": {
        "ProductId": "prod-11111111111111"
      }
    },
    {
      "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": {}
  }
]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

为 SaaS 产品创建具有订阅价格的公开免费试用优惠

以下代码示例演示如何为 SaaS 产品创建具有订阅定价的公共免费试用优惠。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "CreateOffer",
      "Entity": {
        "Type": "Offer@1.0"
      },
      "ChangeName": "CreateOfferChange",
      "DetailsDocument": {
        "ProductId": "prod-111111111111"
      }
    },
    {
      "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": {}
    }
]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

使用合同定价创建替代私募报价

以下代码示例显示了如何根据合同定价的现有协议创建替代私募报价。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "CreateReplacementOffer",
      "Entity": {
        "Type": "Offer@1.0"
      },
      "ChangeName": "CreateReplacementOffer",
      "DetailsDocument": {

```

```

        "AgreementId": "agmt-11111111111111111111111111111111"
    },
    {
        "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": {}
}
]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

描述公开募股

以下代码示例显示了如何描述公开发售。

SDK适用于 Java 2.x

```

// 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;
    }
}
```


- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeEntity](#)”中的。

使私募报价草稿过期

以下代码示例说明如何将私募报价的到期日期设置为过去的某个日期，这样买家就不会再看到该报价。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```
{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "UpdateAvailability",
      "Entity": {
        "Type": "Offer@1.0",
        "Identifier": "offer-11111111111111"
      },
      "DetailsDocument": {
        "AvailabilityEndDate": "2023-01-01"
      }
    }
  ]
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

列出所有私人优惠

以下代码示例显示了如何列出所有私密优惠。

SDK适用于 Java 2.x

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
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.OfferAvailabilityEndDateFilterDateRange;
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;
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

列出针对特定产品 ID 发布的公开和私密报价

以下代码示例展示了如何针对特定产品 ID 列出已发布的公共和私有报价。

SDK适用于 Java 2.x

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
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.getEntitySummaryList());
}
```

```
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;
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

更新报价以应用按使用即用即付定价的合同

以下代码示例显示如何更新报价以应用按使用即用即付定价的合同。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```
{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "UpdatePricingTerms",
      "Entity": {
        "Type": "Offer@1.0",
        "Identifier": "offer-1111111111111111"
      },
      "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"
      }
    }
  ]
}
```



```

    ]
  }
}
]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

更新报价以应用按小时计费的年度定价

以下代码示例显示了如何更新报价以应用按小时计费的年度定价。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "UpdatePricingTerms",
      "Entity": {
        "Type": "Offer@1.0",
        "Identifier": "offer-1111111111111111"
      },
      "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"
                    }
                }
            ]
        }
    ]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

更新报价以将定位应用于特定地理区域

以下代码示例展示了如何更新选件以将定位应用于特定地理区域。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {

```

```

    "ChangeType": "UpdateTargeting",
    "Entity": {
      "Type": "Offer@1.0",
      "Identifier": "offer-111111111111"
    },
    "DetailsDocument": {
      "PositiveTargeting": {
        "CountryCodes": [
          "US",
          "ES",
          "FR",
          "AU"
        ]
      }
    }
  }
]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

更新公开募股的名称和描述

以下代码示例显示如何更新公开发售的名称和描述。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "UpdateLegalTerms",
      "Entity": {
        "Type": "Offer@1.0",
        "Identifier": "offer-111111111111"
      },
      "DetailsDocument": {
        "Terms": [
          {
            "Type": "LegalTerm",

```

```

        "Documents": [
            {
                "Type": "CustomEula",
                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
            }
        ]
    }
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

更新EULA报价

以下代码示例显示了如何更新EULA报价。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "UpdateInformation",
      "Entity": {
        "Type": "Offer@1.0",
        "Identifier": "offer-11111111111111"
      },
      "DetailsDocument": {
        "Name": "New offer name",
        "Description": "New offer description"
      }
    }
  ]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

将私募报价的到期日期更新为 future 日期

以下代码示例说明如何将私募报价的到期日期更新为未来的某个日期，让买家有更多时间评估和接受报价。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```
{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "UpdateAvailability",
      "Entity": {
        "Type": "Offer@1.0",
        "Identifier": "offer-11111111111111"
      },
      "DetailsDocument": {
        "AvailabilityEndDate": "2026-01-01"
      }
    }
  ]
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

更新 SaaS 产品公开免费试用优惠的免费试用时长

以下代码示例显示如何更新 SaaS 产品的公开免费试用优惠的免费试用时长。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```
{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "UpdatePricingTerms",
```

```

    "Entity": {
      "Type": "Offer@1.0",
      "Identifier": "offer-11111111111111"
    },
    "DetailsDocument": {
      "PricingModel": "Usage",
      "Terms": [
        {
          "Type": "FreeTrialPricingTerm",
          "Duration": "P21D",
          "Grants": [
            {
              "DimensionKey": "WorkloadSmall"
            },
            {
              "DimensionKey": "WorkloadMedium"
            }
          ]
        }
      ]
    }
  ]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

更新报价的退款政策

以下代码示例显示了如何更新报价的退款政策。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "UpdateSupportTerms",
      "Entity": {
        "Type": "Offer@1.0",

```

```
        "Identifier": "offer-111111111111"
    },
    "DetailsDocument": {
        "Terms": [
            {
                "Type": "SupportTerm",
                "RefundPolicy": "Updated refund policy description"
            }
        ]
    }
}
]
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

产品

描述AMI、SaaS 或容器产品

以下代码示例说明如何描述AMI、SaaS 或容器产品，以及如何检查它是否包含您想了解的有关该产品的所有信息。

SDK适用于 Java 2.x

```
// 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;
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeEntity](#)”中的。

列出所有 S AMI aaS 或容器产品以及相关的公开报价

以下代码示例显示了如何列出所有 AMI SaaS 或容器产品以及相关的公开报价。

SDK适用于 Java 2.x

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
```



```
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;
}
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [DescribeEntity](#)
 - [ListEntities](#)

转售授权

创建转售授权草稿

以下代码示例展示了如何为任何产品类型创建转售授权草稿，这样您就可以在发布给渠道合作伙伴之前在内部对其进行审查。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```
{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "CreateResaleAuthorization",
      "ChangeName": "ResaleAuthorization",
      "Entity": {
        "Type": "ResaleAuthorization@1.0"
      },
      "DetailsDocument": {
        "ProductId": "prod-111111111111",
        "Name": "TestResaleAuthorization",
        "Description": "Worldwide ResaleAuthorization for Test Product",
        "ResellerAccountId": "111111111111"
      }
    }
  ]
}
```

```
    }  
  }  
]  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

描述转售授权

以下代码示例显示了如何描述转售授权。

SDK适用于 Java 2.x

```
// 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;
}
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeEntity](#)”中的。

通过私下报价发布一次性转售授权

以下代码示例展示了如何发布包含私人报价的一次性转售授权，以便渠道合作伙伴可以使用它来创建渠道合作伙伴私有报价 (CPPO)。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "CreateResaleAuthorization",
      "ChangeName": "ResaleAuthorization",
      "Entity": {
        "Type": "ResaleAuthorization@1.0"
      },
      "DetailsDocument": {

```

```

        "ProductId": "prod-111111111111",
        "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
    }
}
]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

发布带有到期日期的多用途转售授权

以下代码示例展示了如何为按小时按年定价AMI的产品发布带有有效期的多用途转售授权，以便渠道合作伙伴可以使用它来创建 CPPO

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```
{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "CreateResaleAuthorization",
      "ChangeName": "ResaleAuthorization",
      "Entity": {
        "Type": "ResaleAuthorization@1.0"
      },
      "DetailsDocument": {
        "ProductId": "prod-111111111111",
        "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": {}
    }
]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

发布带有到期日期和有效期的多用途转售授权 EULA

以下代码示例展示了如何发布包含任何产品类型的过期日期的多用途转售授权，以及如何添加EULA要发送给买家的自定义授权。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "CreateResaleAuthorization",
      "ChangeName": "ResaleAuthorization",
      "Entity": {
        "Type": "ResaleAuthorization@1.0"
      },
      "DetailsDocument": {
        "ProductId": "prod-111111111111",
        "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"
          }
        ]
      }
    ]
  }
}
]
}
}
]
}
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

发布包含到期日期和经销商合同文档的多用途转售授权

以下代码示例展示了如何发布任何产品类型的多用途转售授权，以及如何为任何产品类型发布具有到期日期的多用途转售授权，以及如何在和渠道合作伙伴之间添加经销商合同文档。ISV

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "CreateResaleAuthorization",
      "ChangeName": "ResaleAuthorization",
      "Entity": {
        "Type": "ResaleAuthorization@1.0"
      },
      "DetailsDocument": {
        "ProductId": "prod-111111111111",
        "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-Contract-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"
      }
    }
  ]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

发布已到期的多用途转售授权，并添加特定的买家账户

以下代码示例展示了如何发布包含任何产品类型的过期日期的多用途转售授权，以及如何为转售添加特定的买家账户。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities `RunChangesets` 中，以从“实用工具”部分启动变更集。

```

{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "CreateResaleAuthorization",
      "ChangeName": "ResaleAuthorization",
      "Entity": {
        "Type": "ResaleAuthorization@1.0"
      },
      "DetailsDocument": {
        "ProductId": "prod-111111111111",
        "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"
          }
        ]
      }
    ]
  }
}

```

```

    ]
  }
}
]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

发布没有到期日期的多用途转售授权

以下代码示例显示了如何为按小时按年定价AMI的产品发布没有到期日期的多用途转售授权，以便CP可以使用该授权来创建. CPPO

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "CreateResaleAuthorization",
      "ChangeName": "ResaleAuthorization",
      "Entity": {
        "Type": "ResaleAuthorization@1.0"
      },
      "DetailsDocument": {
        "ProductId": "prod-111111111111",
        "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"
        }
    ]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

发布多用途转售授权，但没有到期日期和 EULA

以下代码示例展示了如何发布任何产品类型的多用途转售授权，而无需过期日期，以及如何添加EULA要发送给买家的自定义授权。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "CreateResaleAuthorization",
      "ChangeName": "ResaleAuthorization",
      "Entity": {
        "Type": "ResaleAuthorization@1.0"
      },
      "DetailsDocument": {
        "ProductId": "prod-111111111111",
        "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"
                    }
                ]
            }
        ]
    }
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

发布没有到期日期的多用途转售授权和经销商合同文档

以下代码示例展示了如何发布任何产品类型的多用途转售授权（不设到期日期），以及如何添加与渠道合作伙伴之间的经销商合同文档。ISV

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "CreateResaleAuthorization",
      "ChangeName": "ResaleAuthorization",
      "Entity": {
        "Type": "ResaleAuthorization@1.0"
      },
      "DetailsDocument": {
        "ProductId": "prod-11111111111111",

```

```

        "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"
            }
          ]
        }
      ]
    }
  }
]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

发布未过期的多用途转售授权并添加特定的买家账户

以下代码示例展示了如何发布任何产品类型的多用途转售授权，而无需过期日期，以及如何为转售添加特定的买家账户。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```
{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "CreateResaleAuthorization",
      "ChangeName": "ResaleAuthorization",
      "Entity": {
        "Type": "ResaleAuthorization@1.0"
      },
      "DetailsDocument": {
        "ProductId": "prod-111111111111",
        "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"
            }
          ]
        }
      ]
    }
  ]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

发布一次性转售授权并添加灵活的付款时间表

以下代码示例展示了如何发布任何产品类型的一次性转售授权以及如何添加灵活的付款计划。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "CreateResaleAuthorization",
      "ChangeName": "ResaleAuthorization",
      "Entity": {
        "Type": "ResaleAuthorization@1.0"
      },
      "DetailsDocument": {
        "ProductId": "prod-111111111111",
        "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"
                    }
                ]
            }
        ]
    }
}

```

```

    ]
  }
}
]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

发布一次性转售授权并添加 EULA

以下代码示例演示如何发布任何产品类型的一次性转售授权，以及如何添加EULA要发送给买家的自定义授权。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "CreateResaleAuthorization",
      "ChangeName": "ResaleAuthorization",
      "Entity": {
        "Type": "ResaleAuthorization@1.0"
      },
      "DetailsDocument": {
        "ProductId": "prod-11111111111111",
        "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"
            }
          ]
        }
      ]
    }
  ]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

发布一次性转售授权并添加特定的买家账户

以下代码示例展示了如何发布任何产品类型的一次性转售授权，以及如何为转售添加特定的买家账户。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "CreateResaleAuthorization",
      "ChangeName": "ResaleAuthorization",
      "Entity": {
        "Type": "ResaleAuthorization@1.0"
      }
    }
  ]
}

```

```

    "DetailsDocument": {
      "ProductId": "prod-111111111111",
      "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"
                ]
            }
        }
    ]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

发布一次性转售授权并添加经销商合同文档

以下代码示例展示了如何发布任何产品类型的一次性转售授权，以及如何添加与渠道合作伙伴之间的经销商合同文档。ISV

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "CreateResaleAuthorization",
      "ChangeName": "ResaleAuthorization",
      "Entity": {
        "Type": "ResaleAuthorization@1.0"
      },
      "DetailsDocument": {
        "ProductId": "prod-111111111111",
        "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"
          }
        ]
      }
    ]
  }
}
]
}
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

发布一次性转售授权并添加是否续订

以下代码示例显示了如何发布任何产品类型的一次性转售授权以及如何添加是否续订。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```
{
```

```

"Catalog": "AWSMarketplace",
"ChangeSet": [
  {
    "ChangeType": "CreateResaleAuthorization",
    "ChangeName": "ResaleAuthorization",
    "Entity": {
      "Type": "ResaleAuthorization@1.0"
    },
    "DetailsDocument": {
      "ProductId": "prod-111111111111",
      "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"
        }
      }
    }
  ]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

限制转售授权

以下代码示例显示了如何限制转售授权。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "RestrictResaleAuthorization",
      "Entity": {
        "Type": "ResaleAuthorization@1.0",
        "Identifier": "resaleauthz-111111111111"
      },
      "DetailsDocument": {}
    }
  ]
}

```



```
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

更新一次性或多用途转售授权的名称和描述

以下代码示例说明如何在发布任何产品类型之前更新一次性或多用途转售授权的名称和描述。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```
{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "UpdateInformation",
      "Entity": {
        "Type": "ResaleAuthorization@1.0",
        "Identifier": "resaleauthz-11111111111111"
      },
      "DetailsDocument": {
        "Name": "TestResaleAuthorization",
        "Description": "Worldwide ResaleAuthorization for Test Product"
      }
    }
  ]
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

SaaS 产品

使用公开报价草稿创建 SaaS 产品草稿

以下代码示例展示了如何使用公开报价草稿创建 SaaS 产品草稿。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "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"
      }
    }
  ]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

使用合同定价创建公开或有限的 SaaS 产品和公开报价

以下代码示例展示了如何使用合同定价创建公开或有限的 SaaS 产品和公开报价。此示例创建标准或自定义EULA。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 **Utilities RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [

```

```
{
  "ChangeType": "CreateProduct",
  "Entity": {
    "Type": "SaaSProduct@1.0"
  },
  "ChangeName": "CreateProductChange",
  "DetailsDocument": {}
},
{
  "ChangeType": "UpdateInformation",
  "Entity": {
    "Type": "SaaSProduct@1.0",
    "Identifier": "${CreateProductChange.Entity.Identifier}"
  },
  "DetailsDocument": {
    "ProductTitle": "Sample product",
    "ShortDescription": "Brief description",
    "LongDescription": "Detailed description",
    "Highlights": [
      "Sample highlight"
    ],
    "SearchKeywords": [
      "Sample keyword"
    ],
    "Categories": [
      "Data Catalogs"
    ],
    "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
    "VideoUrls": [
      "https://sample.amazonaws.com/awsmvp-video-1"
    ],
    "AdditionalResources": []
  }
},
{
  "ChangeType": "UpdateTargeting",
  "Entity": {
    "Type": "SaaSProduct@1.0",
    "Identifier": "${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": {}
    }
]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

使用即用即付定价合同创建公开或有限的 SaaS 产品和公开报价

以下代码示例演示如何使用即用即付定价的合同创建公开或有限的 SaaS 产品和公开报价。此示例创建标准或自定义EULA。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "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/awssmp-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": {}
}
]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

使用订阅定价创建公开或有限的 SaaS 产品和公开报价

以下代码示例展示了如何使用订阅定价创建公开或有限的 SaaS 产品和公开报价。此示例创建了标准或自定义EULA。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "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/awsmvp-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": {}
  }
]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

发布 SaaS 产品和相关的公开报价

以下代码示例展示了如何发布 SaaS 产品和相关的公开报价。默认情况下，产品将处于受限状态。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "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/awssmp-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": {}
  }
]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

根据现有草稿发布 SaaS 产品和相关的公开报价

以下代码示例展示了如何根据现有草稿发布 SaaS 产品和相关的公开报价。默认情况下，产品将处于受限状态。

SDK适用于 Java 2.x

```

{
  "Catalog": "AWSMarketplace",
  "ChangeSet": [
    {
      "ChangeType": "UpdateVisibility",

```



```

    "ChangeName": "CreateProductChange",
    "Entity": {
      "Type": "SaaSProduct@1.0",
      "Identifier": "prod-111111111111"
    },
    "DetailsDocument": {
      "TargetVisibility": "Public"
    }
  }
]
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

更新AMI或 SaaS 产品的维度

以下代码示例显示了如何更新AMI或 SaaS 产品的维度。

SDK适用于 Java 2.x

要运行此示例，请将以下JSON变更集传递到 Utilities **RunChangesets** 中，以从“实用工具”部分启动变更集。

```

{
  "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"
        }
      ]
    }
  ]
}

```

```
    }  
  ]  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

实用程序

启动变更集的实用程序

以下代码示例显示了如何定义用于启动 AWS Marketplace Catalog API 变更集的实用程序。

SDK适用于 Java 2.x

用于从JSON文件加载变更集并开始处理变更集的实用程序。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
// SPDX-License-Identifier: Apache-2.0  
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 entityIdIdentifier = 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;
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartChangeSet](#)”中的。

MediaConvert 用SDK于 Java 2.x 的示例

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 MediaConvert。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

CreateJob

以下代码示例演示如何使用 CreateJob。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
package com.example.mediaconvert;

import java.net.URI;
import java.util.HashMap;
import java.util.Map;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
```

```
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.Output;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroup;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.HlsGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroupType;
import software.amazon.awssdk.services.mediaconvert.model.HlsDirectoryStructure;
import software.amazon.awssdk.services.mediaconvert.model.HlsManifestDurationFormat;
import software.amazon.awssdk.services.mediaconvert.model.HlsStreamInfResolution;
import software.amazon.awssdk.services.mediaconvert.model.HlsClientCache;
import software.amazon.awssdk.services.mediaconvert.model.HlsCaptionLanguageSetting;
import software.amazon.awssdk.services.mediaconvert.model.HlsManifestCompression;
import software.amazon.awssdk.services.mediaconvert.model.HlsCodecSpecification;
import software.amazon.awssdk.services.mediaconvert.model.HlsOutputSelection;
import software.amazon.awssdk.services.mediaconvert.model.HlsProgramDateTime;
import software.amazon.awssdk.services.mediaconvert.model.HlsTimedMetadataId3Frame;
import software.amazon.awssdk.services.mediaconvert.model.HlsSegmentControl;
import software.amazon.awssdk.services.mediaconvert.model.FileGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.ContainerSettings;
import software.amazon.awssdk.services.mediaconvert.model.VideoDescription;
import software.amazon.awssdk.services.mediaconvert.model.ContainerType;
import software.amazon.awssdk.services.mediaconvert.model.ScalingBehavior;
import software.amazon.awssdk.services.mediaconvert.model.VideoTimecodeInsertion;
import software.amazon.awssdk.services.mediaconvert.model.ColorMetadata;
import software.amazon.awssdk.services.mediaconvert.model.RespondToAfd;
import software.amazon.awssdk.services.mediaconvert.model.AfdSignaling;
import software.amazon.awssdk.services.mediaconvert.model.DropFrameTimecode;
import software.amazon.awssdk.services.mediaconvert.model.VideoCodecSettings;
import software.amazon.awssdk.services.mediaconvert.model.H264Settings;
import software.amazon.awssdk.services.mediaconvert.model.VideoCodec;
import software.amazon.awssdk.services.mediaconvert.model.CreateJobRequest;
import software.amazon.awssdk.services.mediaconvert.model.H264RateControlMode;
import software.amazon.awssdk.services.mediaconvert.model.H264QualityTuningLevel;
import software.amazon.awssdk.services.mediaconvert.model.H264SceneChangeDetect;
import
    software.amazon.awssdk.services.mediaconvert.model.AacAudioDescriptionBroadcasterMix;
import software.amazon.awssdk.services.mediaconvert.model.H264ParControl;
import software.amazon.awssdk.services.mediaconvert.model.AacRawFormat;
import software.amazon.awssdk.services.mediaconvert.model.H264QvbrSettings;
import
    software.amazon.awssdk.services.mediaconvert.model.H264FramerateConversionAlgorithm;
import software.amazon.awssdk.services.mediaconvert.model.H264CodecLevel;
import software.amazon.awssdk.services.mediaconvert.model.H264FramerateControl;
```

```
import software.amazon.awssdk.services.mediaconvert.model.AacCodingMode;
import software.amazon.awssdk.services.mediaconvert.model.H264Telecine;
import
    software.amazon.awssdk.services.mediaconvert.model.H264FlickerAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.H264GopSizeUnits;
import software.amazon.awssdk.services.mediaconvert.model.H264CodecProfile;
import software.amazon.awssdk.services.mediaconvert.model.H264GopBReference;
import software.amazon.awssdk.services.mediaconvert.model.AudioTypeControl;
import software.amazon.awssdk.services.mediaconvert.model.AntiAlias;
import software.amazon.awssdk.services.mediaconvert.model.H264SlowPal;
import
    software.amazon.awssdk.services.mediaconvert.model.H264SpatialAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.H264Syntax;
import software.amazon.awssdk.services.mediaconvert.model.M3u8Settings;
import software.amazon.awssdk.services.mediaconvert.model.InputDenoiseFilter;
import
    software.amazon.awssdk.services.mediaconvert.model.H264TemporalAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.CreateJobResponse;
import
    software.amazon.awssdk.services.mediaconvert.model.H264UnregisteredSeiTimecode;
import software.amazon.awssdk.services.mediaconvert.model.H264EntropyEncoding;
import software.amazon.awssdk.services.mediaconvert.model.InputPsiControl;
import software.amazon.awssdk.services.mediaconvert.model.ColorSpace;
import software.amazon.awssdk.services.mediaconvert.model.H264RepeatPps;
import software.amazon.awssdk.services.mediaconvert.model.H264FieldEncoding;
import software.amazon.awssdk.services.mediaconvert.model.M3u8NielsenId3;
import software.amazon.awssdk.services.mediaconvert.model.InputDeblockFilter;
import software.amazon.awssdk.services.mediaconvert.model.InputRotate;
import software.amazon.awssdk.services.mediaconvert.model.H264DynamicSubGop;
import software.amazon.awssdk.services.mediaconvert.model.TimedMetadata;
import software.amazon.awssdk.services.mediaconvert.model.JobSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioDefaultSelection;
import software.amazon.awssdk.services.mediaconvert.model.VideoSelector;
import software.amazon.awssdk.services.mediaconvert.model.AacSpecification;
import software.amazon.awssdk.services.mediaconvert.model.Input;
import software.amazon.awssdk.services.mediaconvert.model.OutputSettings;
import software.amazon.awssdk.services.mediaconvert.model.H264AdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.AudioLanguageCodeControl;
import software.amazon.awssdk.services.mediaconvert.model.InputFilterEnable;
import software.amazon.awssdk.services.mediaconvert.model.AudioDescription;
import software.amazon.awssdk.services.mediaconvert.model.H264InterlaceMode;
import software.amazon.awssdk.services.mediaconvert.model.AudioCodecSettings;
import software.amazon.awssdk.services.mediaconvert.model.AacSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioCodec;
```



```

import software.amazon.awssdk.services.mediaconvert.model.AacRateControlMode;
import software.amazon.awssdk.services.mediaconvert.model.AacCodecProfile;
import software.amazon.awssdk.services.mediaconvert.model.HlsIFrameOnlyManifest;
import software.amazon.awssdk.services.mediaconvert.model.FrameCaptureSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioSelector;
import software.amazon.awssdk.services.mediaconvert.model.M3u8PcrControl;
import software.amazon.awssdk.services.mediaconvert.model.InputTimecodeSource;
import software.amazon.awssdk.services.mediaconvert.model.HlsSettings;
import software.amazon.awssdk.services.mediaconvert.model.M3u8Scte35Source;

/**
 * Create a MediaConvert job. Must supply MediaConvert access role Amazon
 * Resource Name (ARN), and a
 * valid video input file via Amazon S3 URL.
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateJob {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <mcRoleARN> <fileInput>\s

                Where:
                mcRoleARN - The MediaConvert Role ARN.\s
                fileInput - The URL of an Amazon S3 bucket
                where the input file is located.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String mcRoleARN = args[0];
        String fileInput = args[1];
        Region region = Region.US_WEST_2;
        MediaConvertClient mc = MediaConvertClient.builder()

```

```

        .region(region)
        .build();

        String id = createMediaJob(mc, mcRoleARN, fileInput);
        System.out.println("MediaConvert job created. Job Id = " + id);
        mc.close();
    }

    public static String createMediaJob(MediaConvertClient mc, String mcRoleARN,
String fileInput) {

        String s3path = fileInput.substring(0, fileInput.lastIndexOf('/') +
1) + "javasdk/out/";
        String fileOutput = s3path + "index";
        String thumbsOutput = s3path + "thumbs/";
        String mp4Output = s3path + "mp4/";

        try {
            DescribeEndpointsResponse res = mc

.describeEndpoints(DescribeEndpointsRequest.builder().maxResults(20).build());

            if (res.endpoints().size() <= 0) {
                System.out.println("Cannot find MediaConvert service
endpoint URL!");

                System.exit(1);
            }
            String endpointURL = res.endpoints().get(0).url();
            System.out.println("MediaConvert service URL: " +
endpointURL);

            System.out.println("MediaConvert role arn: " + mcRoleARN);
            System.out.println("MediaConvert input file: " + fileInput);
            System.out.println("MediaConvert output path: " + s3path);

            MediaConvertClient emc = MediaConvertClient.builder()
                .region(Region.US_WEST_2)
                .endpointOverride(URI.create(endpointURL))
                .build();

            // output group Preset HLS low profile
            Output hlsLow = createOutput("hls_low", "_low", "_$dt$",
750000, 7, 1920, 1080, 640);
            // output group Preset HLS media profile

```

```
        Output hlsMedium = createOutput("hls_medium", "_medium", "_$dt$", 1200000, 7, 1920, 1080, 1280);
        // output group Preset HLS high profile
        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();

    Map<String, AudioSelector> audioSelectors = new HashMap<>();
    audioSelectors.put("Audio Selector 1",

AudioSelector.builder().defaultSelection(AudioDefaultSelection.DEFAULT)
        .offset(0).build());

        JobSettings jobSettings =
JobSettings.builder().inputs(Input.builder()
        .audioSelectors(audioSelectors)
        .videoSelector(

VideoSelector.builder().colorSpace(ColorSpace.FOLLOW)

        .rotate(InputRotate.DEGREE_0).build())

        .filterEnable(InputFilterEnable.AUTO).filterStrength(0)
        .deblockFilter(InputDeblockFilter.DISABLED)

        .denoiseFilter(InputDenoiseFilter.DISABLED).psiControl(InputPsiControl.USE_PSI)

        .timecodeSource(InputTimecodeSource.EMBEDDED).fileInput(fileInput).build())
        .outputGroups(appleHLS, thumbs,
fileMp4).build();

        CreateJobRequest createJobRequest =
CreateJobRequest.builder().role(mcRoleARN)
        .settings(jobSettings)
        .build();

        CreateJobResponse createJobResponse =
emc.createJob(createJobRequest);
        return createJobResponse.job().id();

```

```
        } catch (MediaConvertException e) {
            System.out.println(e.toString());
            System.exit(0);
        }
        return "";
    }

    private final static Output createOutput(String customName,
        String nameModifier,
        String segmentModifier,
        int qvbrMaxBitrate,
        int qvbrQualityLevel,
        int originWidth,
        int originHeight,
        int targetWidth) {

        int targetHeight = Math.round(originHeight * targetWidth /
originWidth)
            - (Math.round(originHeight * targetWidth /
originWidth) % 4);
        Output output = null;
        try {
            output =
Output.builder().nameModifier(nameModifier).outputSettings(OutputSettings.builder()

.hlsSettings(HlsSettings.builder().segmentModifier(segmentModifier)

.audioGroupId("program_audio")

.iFrameOnlyManifest(HlsIFrameOnlyManifest.EXCLUDE).build())
                .build())

.containerSettings(ContainerSettings.builder().container(ContainerType.M3_U8)

.m3u8Settings(M3u8Settings.builder().audioFramesPerPes(4)

.pcrControl(M3u8PcrControl.PCR_EVERY_PES_PACKET)

.pmtPid(480).privateMetadataPid(503)

.programNumber(1).patInterval(0).pmtInterval(0)

.scte35Source(M3u8Scte35Source.NONE)
```

```

.scte35Pid(500).nielsenId3(M3u8NielsenId3.NONE)

.timedMetadata(TimedMetadata.NONE)

.timedMetadataPid(502).videoPid(481)

.audioPids(482, 483, 484, 485, 486, 487, 488,
           489, 490, 491, 492)

                                           .build()
                                           .build()
                                           .videoDescription(
VideoDescription.builder().width(targetWidth)

.height(targetHeight)

.scalingBehavior(ScalingBehavior.DEFAULT)

.sharpness(50).antiAlias(AntiAlias.ENABLED)

.timecodeInsertion(
    VideoTimecodeInsertion.DISABLED)

.colorMetadata(ColorMetadata.INSERT)

.respondToAfd(RespondToAfd.NONE)

.afdSignaling(AfdSignaling.NONE)

.dropFrameTimecode(DropFrameTimecode.ENABLED)

.codecSettings(VideoCodecSettings.builder()

    .codec(VideoCodec.H_264)

    .h264Settings(H264Settings

        .builder()

            .rateControlMode(

```

```
                H264RateControlMode.QVBR)

        .parControl(H264ParControl.INITIALIZE_FROM_SOURCE)

        .qualityTuningLevel(

                H264QualityTuningLevel.SINGLE_PASS)

        .qvbrSettings(H264QvbrSettings

                .builder()

                .qvbrQualityLevel(

                        qvbrQualityLevel)

                .build())

        .codecLevel(H264CodecLevel.AUTO)

        .codecProfile((targetHeight > 720

                && targetWidth > 1280)

                ? H264CodecProfile.HIGH

                : H264CodecProfile.MAIN)

        .maxBitrate(qvbrMaxBitrate)

        .framerateControl(

                H264FramerateControl.INITIALIZE_FROM_SOURCE)

        .gopSize(2.0)

        .gopSizeUnits(H264GopSizeUnits.SECONDS)

        .numberBFramesBetweenReferenceFrames(

                2)

        .gopClosedCadence(
```

```
        1)

        .gopBReference(H264GopBReference.DISABLED)

        .slowPal(H264SlowPal.DISABLED)

        .syntax(H264Syntax.DEFAULT)

        .numberReferenceFrames(

            3)

        .dynamicSubGop(H264DynamicSubGop.STATIC)

        .fieldEncoding(H264FieldEncoding.PAFF)

        .sceneChangeDetect(

            H264SceneChangeDetect.ENABLED)

        .minIInterval(0)

        .telecine(H264Telecine.NONE)

        .framerateConversionAlgorithm(

            H264FramerateConversionAlgorithm.DUPLICATE_DROP)

        .entropyEncoding(

            H264EntropyEncoding.CABAC)

        .slices(1)

        .unregisteredSeiTimecode(

            H264UnregisteredSeiTimecode.DISABLED)

        .repeatPps(H264RepeatPps.DISABLED)

        .adaptiveQuantization(

            H264AdaptiveQuantization.HIGH)
```

```
        .spatialAdaptiveQuantization(  
            H264SpatialAdaptiveQuantization.ENABLED)  
        .temporalAdaptiveQuantization(  
            H264TemporalAdaptiveQuantization.ENABLED)  
        .flickerAdaptiveQuantization(  
            H264FlickerAdaptiveQuantization.DISABLED)  
        .softness(0)  
        .interlaceMode(H264InterlaceMode.PROGRESSIVE)  
        .build())  
    .build()                                                                    .build()  
  
    .audioDescriptions(AudioDescription.builder()  
    .audioTypeControl(AudioTypeControl.FOLLOW_INPUT)  
    .languageCodeControl(AudioLanguageCodeControl.FOLLOW_INPUT)  
    .codecSettings(AudioCodecSettings.builder()  
    .codec(AudioCodec.AAC).aacSettings(AacSettings  
        .builder()  
        .codecProfile(AacCodecProfile.LC)  
        .rateControlMode(  
            AacRateControlMode.CBR)  
        .codingMode(AacCodingMode.CODING_MODE_2_0)  
        .sampleRate(44100)
```

```

        .bitrate(96000)

        .rawFormat(AacRawFormat.NONE)

        .specification(AacSpecification.MPEG4)

        .audioDescriptionBroadcasterMix(

            AacAudioDescriptionBroadcasterMix.NORMAL)

        .build()

        .build()

        .build()

        .build();
    } catch (MediaConvertException e) {
        e.printStackTrace();
        System.exit(0);
    }
    return output;
}
}
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateJob](#)”中的。

GetJob

以下代码示例演示如何使用 GetJob。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.GetJobRequest;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;

```



```
import software.amazon.awssdk.services.mediaconvert.model.GetJobResponse;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import java.net.URI;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetJob {

    public static void main(String[] args) {

        final String usage = "\n" +
            " <jobId> \n\n" +
            "Where:\n" +
            " jobId - The job id value.\n\n";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String jobId = args[0];
        Region region = Region.US_WEST_2;
        MediaConvertClient mc = MediaConvertClient.builder()
            .region(region)
            .build();

        getSpecificJob(mc, jobId);
        mc.close();
    }

    public static void getSpecificJob(MediaConvertClient mc, String jobId) {
        try {
            DescribeEndpointsResponse res =
mc.describeEndpoints(DescribeEndpointsRequest.builder()
                .maxResults(20)
                .build());

```

```
        if (res.endpoints().size() <= 0) {
            System.out.println("Cannot find MediaConvert service endpoint
URL!");
            System.exit(1);
        }
        String endpointURL = res.endpoints().get(0).url();
        MediaConvertClient emc = MediaConvertClient.builder()
            .region(Region.US_WEST_2)
            .endpointOverride(URI.create(endpointURL))
            .build();

        GetJobRequest jobRequest = GetJobRequest.builder()
            .id(jobId)
            .build();

        GetJobResponse response = emc.getJob(jobRequest);
        System.out.println("The ARN of the job is " + response.job().arn());

    } catch (MediaConvertException e) {
        System.out.println(e.toString());
        System.exit(0);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetJob](#)”中的。

ListJobs

以下代码示例演示如何使用 ListJobs。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
```

```
import software.amazon.awssdk.services.mediaconvert.model.ListJobsRequest;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.ListJobsResponse;
import software.amazon.awssdk.services.mediaconvert.model.Job;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import java.net.URI;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListJobs {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MediaConvertClient mc = MediaConvertClient.builder()
            .region(region)
            .build();

        listCompleteJobs(mc);
        mc.close();
    }

    public static void listCompleteJobs(MediaConvertClient mc) {
        try {
            DescribeEndpointsResponse res =
mc.describeEndpoints(DescribeEndpointsRequest.builder()
                .maxResults(20)
                .build());

            if (res.endpoints().size() <= 0) {
                System.out.println("Cannot find MediaConvert service endpoint
URL!");
                System.exit(1);
            }

            String endpointURL = res.endpoints().get(0).url();
            MediaConvertClient emc = MediaConvertClient.builder()
                .region(Region.US_WEST_2)
```

```
        .endpointOverride(URI.create(endpointURL))
        .build();

    ListJobsRequest jobsRequest = ListJobsRequest.builder()
        .maxResults(10)
        .status("COMPLETE")
        .build();

    ListJobsResponse jobsResponse = emc.listJobs(jobsRequest);
    List<Job> jobs = jobsResponse.jobs();
    for (Job job : jobs) {
        System.out.println("The JOB ARN is : " + job.arn());
    }

} catch (MediaConvertException e) {
    System.out.println(e.toString());
    System.exit(0);
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListJobs](#)”中的。

使用SDK适用于 Java 2.x 的 Migration Hub 示例

以下代码示例向您展示了如何使用 with Migration Hub 来执行操作和实现常见场景。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

DeleteProgressUpdateStream

以下代码示例演示如何使用 DeleteProgressUpdateStream。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteProgressUpdateStream](#)”中的。

DescribeApplicationState

以下代码示例演示如何使用 DescribeApplicationState。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeApplicationState](#)”中的。

DescribeMigrationTask

以下代码示例演示如何使用 DescribeMigrationTask。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。


```
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);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeMigrationTask](#)”中的。

ImportMigrationTask

以下代码示例演示如何使用 ImportMigrationTask。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ImportMigrationTask](#)”中的。

ListApplications

以下代码示例演示如何使用 ListApplications。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListApplications](#)”中的。

ListCreatedArtifacts

以下代码示例演示如何使用 ListCreatedArtifacts。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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("ProgressStreamB")
        .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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListCreatedArtifacts](#)”中的。

ListMigrationTasks

以下代码示例演示如何使用 ListMigrationTasks。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListMigrationTasks](#)”中的。

Amazon 使用SDK适用于 Java 2.x 的个性化示例

以下代码示例向您展示了如何使用 AWS SDK for Java 2.x 与 Amazon Personalize 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

CreateBatchInferenceJob

以下代码示例演示如何使用 CreateBatchInferenceJob。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateBatchInferenceJob](#)”中的。

CreateCampaign

以下代码示例演示如何使用 CreateCampaign。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void createPersonalCampaign(PersonalizeClient personalizeClient,
String solutionVersionArn,
    String name) {

    try {
        CreateCampaignRequest createCampaignRequest =
CreateCampaignRequest.builder()
            .minProvisionedTPS(1)
            .solutionVersionArn(solutionVersionArn)
            .name(name)
            .build();

        CreateCampaignResponse campaignResponse =
personalizeClient.createCampaign(createCampaignRequest);
        System.out.println("The campaign ARN is " +
campaignResponse.campaignArn());

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateCampaign](#)”中的。

CreateDataset

以下代码示例演示如何使用 CreateDataset。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateDataset](#)”中的。

CreateDatasetExportJob

以下代码示例演示如何使用 CreateDatasetExportJob。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateDatasetExportJob](#)”中的。

CreateDatasetGroup

以下代码示例演示如何使用 CreateDatasetGroup。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。


```
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 "";
}
```

创建域数据集组。

```
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 "";
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateDatasetGroup](#)”中的。

CreateDatasetImportJob

以下代码示例演示如何使用 CreateDatasetImportJob。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateDatasetImportJob](#)”中的。

CreateEventTracker

以下代码示例演示如何使用 CreateEventTracker。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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;
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateEventTracker](#)”中的。

CreateFilter

以下代码示例演示如何使用 CreateFilter。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateFilter](#)”中的。

CreateRecommender

以下代码示例演示如何使用 CreateRecommender。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateRecommender](#)”中的。

CreateSchema

以下代码示例演示如何使用 CreateSchema。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";
}
```

使用域创建架构。


```
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 "";
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateSchema](#)”中的。

CreateSolution

以下代码示例演示如何使用 CreateSolution。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateSolution](#)”中的。

CreateSolutionVersion

以下代码示例演示如何使用 CreateSolutionVersion。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";  
    }
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateSolutionVersion](#)”中的。

DeleteCampaign

以下代码示例演示如何使用 DeleteCampaign。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void deleteSpecificCampaign(PersonalizeClient personalizeClient,  
String campaignArn) {  
  
    try {  
        DeleteCampaignRequest campaignRequest = DeleteCampaignRequest.builder()  
            .campaignArn(campaignArn)  
            .build();  
  
        personalizeClient.deleteCampaign(campaignRequest);  
  
    } catch (PersonalizeException e) {  
        System.err.println(e.awsErrorDetails().errorMessage());  
        System.exit(1);  
    }  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteCampaign](#)”中的。

DeleteEventTracker

以下代码示例演示如何使用 DeleteEventTracker。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void deleteEventTracker(PersonalizeClient personalizeClient,
String eventTrackerArn) {
    try {
        DeleteEventTrackerRequest deleteEventTrackerRequest =
DeleteEventTrackerRequest.builder()
            .eventTrackerArn(eventTrackerArn)
            .build();

        int status =
personalizeClient.deleteEventTracker(deleteEventTrackerRequest).sdkHttpResponse().statusCode();

        System.out.println("Status code:" + status);

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteEventTracker](#)”中的。

DeleteSolution

以下代码示例演示如何使用 DeleteSolution。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteSolution](#)”中的。

DescribeCampaign

以下代码示例演示如何使用 DescribeCampaign。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeCampaign](#)”中的。

DescribeRecipe

以下代码示例演示如何使用 DescribeRecipe。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeRecipe](#)”中的。

DescribeSolution

以下代码示例演示如何使用 DescribeSolution。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeSolution](#)”中的。

ListCampaigns

以下代码示例演示如何使用 ListCampaigns。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListCampaigns](#)”中的。

ListDatasetGroups

以下代码示例演示如何使用 ListDatasetGroups。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListDatasetGroups](#)”中的。

ListRecipes

以下代码示例演示如何使用 ListRecipes。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。


```
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);  
    }  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListRecipes](#)”中的。

ListSolutions

以下代码示例演示如何使用 ListSolutions。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListSolutions](#)”中的。

UpdateCampaign

以下代码示例演示如何使用 UpdateCampaign。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";  
    }
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [UpdateCampaign](#)”中的。

使用SDK适用于 Java 2.x 的亚马逊个性化事件示例

以下代码示例向您展示了如何使用 AWS SDK for Java 2.x 与 Amazon Personalize Events 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

PutEvents

以下代码示例演示如何使用 PutEvents。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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;
}
```


- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [PutEvents](#)”中的。

PutUsers

以下代码示例演示如何使用 PutUsers。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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;
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [PutUsers](#)”中的。

使用 Java 2.x 的 Amazon Personalize SDK 运行时示例

以下代码示例向您展示了如何使用 AWS SDK for Java 2.x 与 Amazon Personalize Runtime 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

GetPersonalizedRanking

以下代码示例演示如何使用 GetPersonalizedRanking。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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;
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetPersonalizedRanking](#)”中的。

GetRecommendations

以下代码示例演示如何使用 GetRecommendations。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

获取推荐项目列表。

```
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);
    }
}

```

从在域数据集组中创建的推荐系统获取推荐项目列表。

```

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);
    }
}

```

请求推荐时使用筛选条件。

```

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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetRecommendations](#)”中的。

使用SDK适用于 Java 2.x 的 Amazon Pinpoint 示例

以下代码示例向您展示了如何使用 AWS SDK for Java 2.x 与 Amazon Pinpoint 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

CreateApp

以下代码示例演示如何使用 CreateApp。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateApp](#)”中的。

CreateCampaign

以下代码示例演示如何使用 CreateCampaign。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

创建市场活动。

```
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;
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateCampaign](#)”中的。

CreateExportJob

以下代码示例演示如何使用 CreateExportJob。

SDK适用于 Java 2.x

Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

导出端点。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateExportJob](#)”中的。

CreateImportJob

以下代码示例演示如何使用 CreateImportJob。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

导入分段。

```
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;
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateImportJob](#)”中的。

CreateSegment

以下代码示例演示如何使用 CreateSegment。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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;
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateSegment](#)”中的。

DeleteApp

以下代码示例演示如何使用 DeleteApp。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

删除应用程序。

```
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);
        }
    }
}
```



```
    }  
  }  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteApp](#)”中的。

DeleteEndpoint

以下代码示例演示如何使用 DeleteEndpoint。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

删除端点。

```
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();

    deletePinEndpoint(pinpoint, appId, endpointId);
    pinpoint.close();
}

public static void deletePinEndpoint(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");
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteEndpoint](#)”中的。

GetEndpoint

以下代码示例演示如何使用 GetEndpoint。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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");
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetEndpoint](#)”中的。

GetSegments

以下代码示例演示如何使用 GetSegments。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

列出分段。

```
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("Segment " + segment.id() + " " + segment.name() +
                    " " + segment.lastModifiedDate());
        }

    } catch (PinpointException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetSegments](#)”中的。

GetSmsChannel

以下代码示例演示如何使用 GetSmsChannel。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetSmsChannel](#)”中的。

GetUserEndpoints

以下代码示例演示如何使用 GetUserEndpoints。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetUserEndpoints](#)”中的。

SendMessages

以下代码示例演示如何使用 SendMessages。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

发送电子邮件。

```
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);
    }
}
}

```

使用 CC 值发送电子邮件。

```

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();
    }
}
}
```

发送一条SMS消息。

```
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.sendMessage(request);
        MessageResponse msg1 = response.getMessageResponse();
        Map map1 = msg1.getResult();

        // Write out the result of sendMessage.
        map1.forEach((k, v) -> System.out.println((k + ":" + v)));

    } catch (PinpointException e) {
        System.err.println(e.getAwsErrorDetails().getErrorMessage());
        System.exit(1);
    }
}
}
}

```

发送批量SMS消息。

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.SMSMessage;

```

```
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesResponse;
import software.amazon.awssdk.services.pinpoint.model.MessageResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessageBatch {

    // The type of SMS message that you want to send. If you plan to send
    // time-sensitive content, specify TRANSACTIONAL. If you plan to send
    // marketing-related content, specify PROMOTIONAL.
    public static String messageType = "TRANSACTIONAL";

    // The registered keyword associated with the originating short code.
    public static String registeredKeyword = "myKeyword";

    // The sender ID to use when sending the message. Support for sender ID
    // varies by country or region. For more information, see
    // https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-countries.html
    public static String senderId = "MySenderId";

    public static void main(String[] args) {
        final String usage = ""

            Usage:  <message> <appId> <originationNumber>
<destinationNumber> <destinationNumber1>\s

            Where:
                message - The body of the message to send.
                appId - The Amazon Pinpoint project/application ID
to use when you send this message.
```

originationNumber - The phone number or short code that you specify has to be associated with your Amazon Pinpoint account. For best results, specify long codes in E.164 format (for example, +1-555-555-5654).

destinationNumber - The recipient's phone number. For best results, you should specify the phone number in E.164 format (for example, +1-555-555-5654).

destinationNumber1 - The second recipient's phone number. For best results, you should specify the phone number in E.164 format (for example, +1-555-555-5654).\s

```

        """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String appId = args[1];
        String originationNumber = args[2];
        String destinationNumber = args[3];
        String destinationNumber1 = args[4];
        System.out.println("Sending a message");
        PinpointClient pinpoint = PinpointClient.builder()
            .region(Region.US_EAST_1)
            .build();

        sendSMSMessage(pinpoint, message, appId, originationNumber,
            destinationNumber, destinationNumber1);
        pinpoint.close();
    }

    public static void sendSMSMessage(PinpointClient pinpoint, String message,
        String appId,
            String originationNumber,
            String destinationNumber, String destinationNumber1) {
        try {
            Map<String, AddressConfiguration> addressMap = new
            HashMap<String, AddressConfiguration>();
            AddressConfiguration addConfig =
            AddressConfiguration.builder()
                .channelType(ChannelType.SMS)
                .build();

```

```
        // Add an entry to the Map object for each number to whom
you want to send a
        // message.
        addressMap.put(destinationNumber, addConfig);
        addressMap.put(destinationNumber1, addConfig);
        SMSMessage smsMessage = SMSMessage.builder()
            .body(message)
            .messageType(messageType)
            .originationNumber(originationNumber)
            .senderId(senderId)
            .keyword(registeredKeyword)
            .build();

        // Create a DirectMessageConfiguration object.
        DirectMessageConfiguration direct =
DirectMessageConfiguration.builder()
            .smsMessage(smsMessage)
            .build();

        MessageRequest msgReq = MessageRequest.builder()
            .addresses(addressMap)
            .messageConfiguration(direct)
            .build();

        // Create a SendMessagesRequest object.
        SendMessagesRequest request = SendMessagesRequest.builder()
            .applicationId(appId)
            .messageRequest(msgReq)
            .build();

        SendMessagesResponse response =
pinpoint.sendMessage(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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [SendMessages](#)”中的。

UpdateEndpoint

以下代码示例演示如何使用 UpdateEndpoint。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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;
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [UpdateEndpoint](#)”中的。

SDK适用于 Java 2.x 的 Amazon Pinpoint SMS 和 Voice API 示例

以下代码示例向您展示了如何使用 AWS SDK for Java 2.x 与 Amazon Pinpoint 和 Voice 配合使用来执行操作SMS和实现常见场景。API

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

SendVoiceMessage

以下代码示例演示如何使用 SendVoiceMessage。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpointsmsvoice.PinpointSmsVoiceClient;
```

```
import software.amazon.awssdk.services.pinpointsmsvoice.model.SSMLMessageType;
import software.amazon.awssdk.services.pinpointsmsvoice.model.VoiceMessageContent;
import
    software.amazon.awssdk.services.pinpointsmsvoice.model.SendVoiceMessageRequest;
import
    software.amazon.awssdk.services.pinpointsmsvoice.model.PinpointSmsVoiceException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendVoiceMessage {

    // The Amazon Polly voice that you want to use to send the message. For a
    list
    // of voices, see https://docs.aws.amazon.com/polly/latest/dg/voicelist.html
    static final String voiceName = "Matthew";

    // The language to use when sending the message. For a list of supported
    // languages, see
    // https://docs.aws.amazon.com/polly/latest/dg/SupportedLanguage.html
    static final String languageCode = "en-US";

    // The content of the message. This example uses SSML to customize and
    control
    // certain aspects of the message, such as by adding pauses and changing
    // phonation. The message can't contain any line breaks.
    static final String ssmlMessage = "<speaK>This is a test message sent from "
        + "<emphasis>Amazon Pinpoint</emphasis> "
        + "using the <break strength='weak'/>AWS "
        + "SDK for Java. "
        + "<amazon:effect phonation='soft'>Thank "
        + "you for listening.</amazon:effect></speaK>";

    public static void main(String[] args) {
```

```
final String usage = ""

        Usage:  <originationNumber> <destinationNumber>\s

        Where:
            originationNumber - The phone number or short code
that you specify has to be associated with your Amazon Pinpoint account. For best
results, specify long codes in E.164 format (for example, +1-555-555-5654).
            destinationNumber - The recipient's phone number.
For best results, you should specify the phone number in E.164 format (for example,
+1-555-555-5654).\s

        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String originationNumber = args[0];
    String destinationNumber = args[1];
    System.out.println("Sending a voice message");

    // Set the content type to application/json.
    List<String> listVal = new ArrayList<>();
    listVal.add("application/json");
    Map<String, List<String>> values = new HashMap<>();
    values.put("Content-Type", listVal);

    ClientOverrideConfiguration config2 =
ClientOverrideConfiguration.builder()
        .headers(values)
        .build();

    PinpointSmsVoiceClient client = PinpointSmsVoiceClient.builder()
        .overrideConfiguration(config2)
        .region(Region.US_EAST_1)
        .build();

    sendVoiceMsg(client, originationNumber, destinationNumber);
    client.close();
}

    public static void sendVoiceMsg(PinpointSmsVoiceClient client, String
originationNumber,
```

```
        String destinationNumber) {
    try {
        SSMLMessageType ssmlMessageType = SSMLMessageType.builder()
            .languageCode(languageCode)
            .text(ssmlMessage)
            .voiceId(voiceName)
            .build();

        VoiceMessageContent content = VoiceMessageContent.builder()
            .ssmlMessage(ssmlMessageType)
            .build();

        SendVoiceMessageRequest voiceMessageRequest =
SendVoiceMessageRequest.builder()
            .destinationPhoneNumber(destinationNumber)
            .originationPhoneNumber(originationNumber)
            .content(content)
            .build();

        client.sendVoiceMessage(voiceMessageRequest);
        System.out.println("The message was sent successfully.");

    } catch (PinpointSmsVoiceException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [SendVoiceMessage](#)”中的。

使用SDK适用于 Java 2.x 的 Amazon Polly 示例

以下代码示例向您展示了如何在 Amazon Polly 中使用来执行操作和实现常见场景。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

DescribeVoices

以下代码示例演示如何使用 DescribeVoices。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeVoices](#)”中的。

ListLexicons

以下代码示例演示如何使用 ListLexicons。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。


```
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);
        }
    }
}
```

```
    }  
  }  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListLexicons](#)”中的。

SynthesizeSpeech

以下代码示例演示如何使用 SynthesizeSpeech。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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;
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [SynthesizeSpeech](#)”中的。

使用SDK适用于 Java 2.x 的亚马逊RDS示例

以下代码示例向您展示如何在 Amazon 中使用来执行操作和实现常见场景RDS。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

开始使用

你好 Amazon RDS

以下代码示例展示了如何开始使用 Amazon RDS。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅describeDBInstances 《AWS SDK for Java 2.x API参考资料》中的 [D](#)。

主题

- [操作](#)
- [场景](#)
- [无服务器示例](#)

操作

CreateDBInstance

以下代码示例演示如何使用 CreateDBInstance。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import com.google.gson.Gson;
import
    software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For more details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating\_how-services-use-secrets\_RS.html
 */

public class CreateDBInstance {
    public static long sleepTime = 20;

    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <dbInstanceIdentifier> <dbName> <secretName>
```

```
        Where:
            dbInstanceIdentifier - The database instance identifier.\s
            dbName - The database name.\s
            secretName - The name of the AWS Secrets Manager secret that
contains the database credentials."
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String dbInstanceIdentifier = args[0];
    String dbName = args[1];
    String secretName = args[2];
    Gson gson = new Gson();
    User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
    Region region = Region.US_WEST_2;
    RdsClient rdsClient = RdsClient.builder()
        .region(region)
        .build();

    createDatabaseInstance(rdsClient, dbInstanceIdentifier, dbName,
user.getUsername(), user.getPassword());
    waitForInstanceReady(rdsClient, dbInstanceIdentifier);
    rdsClient.close();
}

private static SecretsManagerClient getSecretClient() {
    Region region = Region.US_WEST_2;
    return SecretsManagerClient.builder()
        .region(region)
        .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
        .build();
}

private static String getSecretValues(String secretName) {
    SecretsManagerClient secretClient = getSecretClient();
    GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
        .secretId(secretName)
        .build();
```



```
        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    public static void createDatabaseInstance(RdsClient rdsClient,
        String dbInstanceIdentifier,
        String dbName,
        String userName,
        String userPassword) {

        try {
            CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceIdentifier)
                .allocatedStorage(100)
                .dbName(dbName)
                .engine("mysql")
                .dbInstanceClass("db.m4.large")
                .engineVersion("8.0")
                .storageType("standard")
                .masterUsername(userName)
                .masterUserPassword(userPassword)
                .build();

            CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
            System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
```

```
DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
    .dbInstanceIdentifier(dbInstanceIdentifier)
    .build();

// Loop until the cluster is ready.
while (!instanceReady) {
    DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
    List<DBInstance> instanceList = response.dbInstances();
    for (DBInstance instance : instanceList) {
        instanceReadyStr = instance.dbInstanceStatus();
        if (instanceReadyStr.contains("available"))
            instanceReady = true;
        else {
            System.out.print(".");
            Thread.sleep(sleepTime * 1000);
        }
    }
}
System.out.println("Database instance is available!");

} catch (RdsException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- 有关API详细信息，请参阅createDBInstance 《AWS SDK for Java 2.x API参考资料》中的 [C](#)。

CreateDBParameterGroup

以下代码示例演示如何使用 CreateDBParameterGroup。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考资料”中的 [CreateDBParameterGroup](#)。

CreateDBSnapshot

以下代码示例演示如何使用 CreateDBSnapshot。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
// 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);
    }
}

```

- 有关API详细信息，请参阅createDBSnapshot 《AWS SDK for Java 2.x API参考资料》中的 [C](#)。

DeleteDBInstance

以下代码示例演示如何使用 DeleteDBInstance。

SDK适用于 Java 2.x

Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```

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);
        }
    }
}
```

- 有关API详细信息，请参阅deleteDBInstance 《AWS SDK for Java 2.x API参考资料》中的 [D](#)。

DeleteDBParameterGroup

以下代码示例演示如何使用 DeleteDBParameterGroup。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
// 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);
    }
}


```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考”中的 [DeleteDBParameter 组](#)。

DescribeAccountAttributes

以下代码示例演示如何使用 DescribeAccountAttributes。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);  
    }  
}  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeAccountAttributes](#)”中的。

DescribeDBEngineVersions

以下代码示例演示如何使用 DescribeDBEngineVersions。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "  

```

```
        + engine0b.dbParameterGroupFamily());
        System.out.println("The name of the database engine " +
engine0b.engine());
        System.out.println("The version number of the database engine " +
engine0b.engineVersion());
    }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅AWS SDK for Java 2.x API参考中的 [DescribeDBEngine 版本](#)。

DescribeDBInstances

以下代码示例演示如何使用 DescribeDBInstances。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关API详细信息，请参阅describeDBInstances 《AWS SDK for Java 2.x API参考资料》中的 [D](#)。

DescribeDBParameterGroups

以下代码示例演示如何使用 DescribeDBParameterGroups。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考”中的 [DescribeDBParameter 组](#)。

DescribeDBParameters

以下代码示例演示如何使用 DescribeDBParameters。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

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

- 有关API详细信息，请参阅describeDBParameters 《AWS SDK for Java 2.x API参考资料》中的 [D](#)。

DescribeOrderableDBInstanceOptions

以下代码示例演示如何使用 DescribeOrderableDBInstanceOptions。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

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

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeOrderableDBInstanceOptions](#)”中的。

GenerateRDSAuthToken

以下代码示例演示如何使用 GenerateRDSAuthToken。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用 [RdsUtilities](#) 类生成身份验证令牌。

```
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 "";
}
}
```

- 有关API详细信息，请参阅《AWS SDK for Java 2.x API参考资料》中的 [GenerateRDSAuthToken](#)。

ModifyDBInstance

以下代码示例演示如何使用 ModifyDBInstance。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅 `modifyDBInstance` 《AWS SDK for Java 2.x API参考资料》中的 [M](#)。

ModifyDBParameterGroup

以下代码示例演示如何使用 `ModifyDBParameterGroup`。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

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

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考资料”中的 [ModifyDBParameter Group](#)。

RebootDBInstance

以下代码示例演示如何使用 RebootDBInstance。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关API详细信息，请参阅rebootDBInstance 《AWS SDK for Java 2.x API参考资料》中的 [R](#)。


场景

开始使用数据库实例

以下代码示例演示了操作流程：

- 创建自定义数据库参数组并设置参数值。
- 创建一个配置为使用参数组的数据库实例。数据库实例还包含一个数据库。
- 拍摄实例的快照。
- 删除实例和参数组。

SDK适用于 Java 2.x

 Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

运行多个操作。

```
import com.google.gson.Gson;
import
    software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.CreateDbParameterGroupResponse;
import software.amazon.awssdk.services.rds.model.CreateDbSnapshotRequest;
import software.amazon.awssdk.services.rds.model.CreateDbSnapshotResponse;
import software.amazon.awssdk.services.rds.model.DBEngineVersion;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.rds.model.DBParameterGroup;
import software.amazon.awssdk.services.rds.model.DBSnapshot;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbEngineVersionsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbEngineVersionsResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbParameterGroupsResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbParametersResponse;
import software.amazon.awssdk.services.rds.model.DescribeDbSnapshotsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbSnapshotsResponse;
import
    software.amazon.awssdk.services.rds.model.DescribeOrderableDbInstanceOptionsResponse;
import software.amazon.awssdk.services.rds.model.ModifyDbParameterGroupResponse;
import software.amazon.awssdk.services.rds.model.OrderableDBInstanceOption;
import software.amazon.awssdk.services.rds.model.Parameter;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.CreateDbParameterGroupRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbParameterGroupsRequest;
import software.amazon.awssdk.services.rds.model.DescribeDbParametersRequest;
```

```
import software.amazon.awssdk.services.rds.model.ModifyDbParameterGroupRequest;
import
    software.amazon.awssdk.services.rds.model.DescribeOrderableDbInstanceOptionsRequest;
import software.amazon.awssdk.services.rds.model.DeleteDbParameterGroupRequest;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For details, see:
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating\_how-services-use-secrets\_RS.html
 *
 * This Java example performs these tasks:
 *
 * 1. Returns a list of the available DB engines.
 * 2. Selects an engine family and create a custom DB parameter group.
 * 3. Gets the parameter groups.
 * 4. Gets parameters in the group.
 * 5. Modifies the auto_increment_offset parameter.
 * 6. Gets and displays the updated parameters.
 * 7. Gets a list of allowed engine versions.
 * 8. Gets a list of micro instance classes available for the selected engine.
 * 9. Creates an RDS database instance that contains a MySQL database and uses
 * the parameter group.
 * 10. Waits for the DB instance to be ready and prints out the connection
 * endpoint value.
 * 11. Creates a snapshot of the DB instance.
 * 12. Waits for an RDS DB snapshot to be ready.
 * 13. Deletes the RDS DB instance.
 * 14. Deletes the parameter group.
 */
```

```

public class RDSScenario {
    public static long sleepTime = 20;
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = ""

            Usage:
                <dbGroupName> <dbParameterGroupFamily> <dbInstanceIdentifier>
<dbName> <dbSnapshotIdentifier> <secretName>

            Where:
                dbGroupName - The database group name.\s
                dbParameterGroupFamily - The database parameter group name (for
example, mysql8.0).
                dbInstanceIdentifier - The database instance identifier\s
                dbName - The database name.\s
                dbSnapshotIdentifier - The snapshot identifier.\s
                secretName - The name of the AWS Secrets Manager secret that
contains the database credentials"
                """;

        if (args.length != 6) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbGroupName = args[0];
        String dbParameterGroupFamily = args[1];
        String dbInstanceIdentifier = args[2];
        String dbName = args[3];
        String dbSnapshotIdentifier = args[4];
        String secretName = args[5];

        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
User.class);
        String masterUsername = user.getUsername();
        String masterUserPassword = user.getPassword();

        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
            .region(region)
            .build();

```



```
System.out.println(DASHES);
System.out.println("Welcome to the Amazon RDS example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Return a list of the available DB engines");
describeDBEngines(rdsClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Create a custom parameter group");
createDBParameterGroup(rdsClient, dbGroupName, dbParameterGroupFamily);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Get the parameter group");
describeDbParameterGroups(rdsClient, dbGroupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Get the parameters in the group");
describeDbParameters(rdsClient, dbGroupName, 0);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Modify the auto_increment_offset parameter");
modifyDBParas(rdsClient, dbGroupName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Display the updated value");
describeDbParameters(rdsClient, dbGroupName, -1);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Get a list of allowed engine versions");
getAllowedEngines(rdsClient, dbParameterGroupFamily);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Get a list of micro instance classes available for
the selected engine");
getMicroInstances(rdsClient);
System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println(
            "9. Create an RDS database instance that contains a MySQL database
and uses the parameter group");
        String dbARN = createDatabaseInstance(rdsClient, dbGroupName,
dbInstanceIdentifier, dbName, masterUsername,
            masterUserPassword);
        System.out.println("The ARN of the new database is " + dbARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Wait for DB instance to be ready");
        waitForInstanceReady(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Create a snapshot of the DB instance");
        createSnapshot(rdsClient, dbInstanceIdentifier, dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Wait for DB snapshot to be ready");
        waitForSnapshotReady(rdsClient, dbInstanceIdentifier, dbSnapshotIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Delete the DB instance");
        deleteDatabaseInstance(rdsClient, dbInstanceIdentifier);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Delete the parameter group");
        deleteParaGroup(rdsClient, dbGroupName, dbARN);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);

        rdsClient.close();
    }

    private static SecretsManagerClient getSecretClient() {
```

```
        Region region = Region.US_WEST_2;
        return SecretsManagerClient.builder()
            .region(region)

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

    public static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
            .secretId(secretName)
            .build();

        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    // Delete the parameter group after database has been deleted.
    // An exception is thrown if you attempt to delete the para group while database
    // exists.
    public static void deleteParaGroup(RdsClient rdsClient, String dbGroupName,
String dbARN)
        throws InterruptedException {
        try {
            boolean isDataDel = false;
            boolean didFind;
            String instanceARN;

            // Make sure that the database has been deleted.
            while (!isDataDel) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
                List<DBInstance> instanceList = response.dbInstances();
                int listSize = instanceList.size();
                didFind = false;
                int index = 1;
                for (DBInstance instance : instanceList) {
                    instanceARN = instance.dbInstanceArn();
                    if (instanceARN.compareTo(dbARN) == 0) {
                        System.out.println(dbARN + " still exists");
                        didFind = true;
                    }
                }
            }
        }
    }
}
```

```
        if ((index == listSize) && (!didFind)) {
            // Went through the entire list and did not find the
database ARN.
            isDataDel = true;
        }
        Thread.sleep(sleepTime * 1000);
        index++;
    }
}

// Delete the para group.
DeleteDbParameterGroupRequest parameterGroupRequest =
DeleteDbParameterGroupRequest.builder()
    .dbParameterGroupName(dbGroupName)
    .build();

rdsClient.deleteDBParameterGroup(parameterGroupRequest);
System.out.println(dbGroupName + " was deleted.");

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}

// Delete the DB instance.
public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
    try {
        DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .deleteAutomatedBackups(true)
            .skipFinalSnapshot(true)
            .build();

        DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
        System.out.print("The status of the database is " +
response.dbInstance().dbInstanceStatus());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

```
    }  
  }  
  
  // Waits until the snapshot instance is available.  
  public static void waitForSnapshotReady(RdsClient rdsClient, String  
dbInstanceIdentifier,  
    String dbSnapshotIdentifier) {  
    try {  
      boolean snapshotReady = false;  
      String snapshotReadyStr;  
      System.out.println("Waiting for the snapshot to become available.");  
  
      DescribeDbSnapshotsRequest snapshotsRequest =  
DescribeDbSnapshotsRequest.builder()  
        .dbSnapshotIdentifier(dbSnapshotIdentifier)  
        .dbInstanceIdentifier(dbInstanceIdentifier)  
        .build();  
  
      while (!snapshotReady) {  
        DescribeDbSnapshotsResponse response =  
rdsClient.describeDBSnapshots(snapshotsRequest);  
        List<DBSnapshot> snapshotList = response.dbSnapshots();  
        for (DBSnapshot snapshot : snapshotList) {  
          snapshotReadyStr = snapshot.status();  
          if (snapshotReadyStr.contains("available")) {  
            snapshotReady = true;  
          } else {  
            System.out.print(".");  
            Thread.sleep(sleepTime * 1000);  
          }  
        }  
      }  
  
      System.out.println("The Snapshot is available!");  
    } catch (RdsException | InterruptedException e) {  
      System.out.println(e.getLocalizedMessage());  
      System.exit(1);  
    }  
  }  
  
  // Create an Amazon RDS snapshot.  
  public static void createSnapshot(RdsClient rdsClient, String  
dbInstanceIdentifier, String dbSnapshotIdentifier) {  
    try {
```

```
        CreateDbSnapshotRequest snapshotRequest =
CreateDbSnapshotRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .dbSnapshotIdentifier(dbSnapshotIdentifier)
            .build();

        CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
        System.out.println("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Waits until the database instance is available.
public static void waitForInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
    boolean instanceReady = false;
    String instanceReadyStr;
    System.out.println("Waiting for instance to become available.");
    try {
        DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .build();

        String endpoint = "";
        while (!instanceReady) {
            DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
            List<DBInstance> instanceList = response.dbInstances();
            for (DBInstance instance : instanceList) {
                instanceReadyStr = instance.dbInstanceStatus();
                if (instanceReadyStr.contains("available")) {
                    endpoint = instance.endpoint().address();
                    instanceReady = true;
                } else {
                    System.out.print(".");
                    Thread.sleep(sleepTime * 1000);
                }
            }
        }
    }
```

```
    }
    System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

    } catch (RdsException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

// Create a database instance and return the ARN of the database.
public static String createDatabaseInstance(RdsClient rdsClient,
    String dbGroupName,
    String dbInstanceIdentifier,
    String dbName,
    String masterUsername,
    String masterUserPassword) {

    try {
        CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
            .dbInstanceIdentifier(dbInstanceIdentifier)
            .allocatedStorage(100)
            .dbName(dbName)
            .dbParameterGroupName(dbGroupName)
            .engine("mysql")
            .dbInstanceClass("db.m4.large")
            .engineVersion("8.0")
            .storageType("standard")
            .masterUsername(masterUsername)
            .masterUserPassword(masterUserPassword)
            .build();

        CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
        System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
        return response.dbInstance().dbInstanceArn();

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}
```

```
        return "";
    }

    // Get a list of micro instances.
    public static void getMicroInstances(RdsClient rdsClient) {
        try {
            DescribeOrderableDbInstanceOptionsRequest dbInstanceOptionsRequest =
DescribeOrderableDbInstanceOptionsRequest
                .builder()
                .engine("mysql")
                .build();

            DescribeOrderableDbInstanceOptionsResponse response = rdsClient
                .describeOrderableDBInstanceOptions(dbInstanceOptionsRequest);
            List<OrderableDBInstanceOption> orderableDBInstances =
response.orderableDBInstanceOptions();
            for (OrderableDBInstanceOption dbInstanceOption : orderableDBInstances)
            {
                System.out.println("The engine version is " +
dbInstanceOption.engineVersion());
                System.out.println("The engine description is " +
dbInstanceOption.engine());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Get a list of allowed engine versions.
    public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
        try {
            DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
                .dbParameterGroupFamily(dbParameterGroupFamily)
                .engine("mysql")
                .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
            List<DBEngineVersion> dbEngines = response.dbEngineVersions();
            for (DBEngineVersion dbEngine : dbEngines) {
```



```
        System.out.println("The engine version is " +
dbEngine.engineVersion());
        System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
    }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Modify auto_increment_offset and auto_increment_increment parameters.
public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
    try {
        Parameter parameter1 = Parameter.builder()
            .parameterName("auto_increment_offset")
            .applyMethod("immediate")
            .parameterValue("5")
            .build();

        List<Parameter> paraList = new ArrayList<>();
        paraList.add(parameter1);
        ModifyDbParameterGroupRequest groupRequest =
ModifyDbParameterGroupRequest.builder()
            .dbParameterGroupName(dbGroupName)
            .parameters(paraList)
            .build();

        ModifyDbParameterGroupResponse response =
rdsClient.modifyDBParameterGroup(groupRequest);
        System.out.println("The parameter group " +
response.dbParameterGroupName() + " was successfully modified");

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

// Retrieve parameters in the group.
public static void describeDbParameters(RdsClient rdsClient, String dbGroupName,
int flag) {
    try {
```

```

        DescribeDbParametersRequest dbParameterGroupsRequest;
        if (flag == 0) {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .build();
        } else {
            dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .source("user")
                .build();
        }

        DescribeDbParametersResponse response =
rdsClient.describeDBParameters(dbParameterGroupsRequest);
        List<Parameter> dbParameters = response.parameters();
        String paraName;
        for (Parameter para : dbParameters) {
            // Only print out information about either auto_increment_offset or
            // auto_increment_increment.
            paraName = para.parameterName();
            if ((paraName.compareTo("auto_increment_offset") == 0)
                || (paraName.compareTo("auto_increment_increment ") == 0)) {
                System.out.println("*** The parameter name is " + paraName);
                System.out.println("*** The parameter value is " +
para.parameterValue());
                System.out.println("*** The parameter data type is " +
para.dataType());
                System.out.println("*** The parameter description is " +
para.description());
                System.out.println("*** The parameter allowed values is " +
para.allowedValues());
            }
        }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDbParameterGroups(RdsClient rdsClient, String
dbGroupName) {
        try {

```

```
        DescribeDbParameterGroupsRequest groupsRequest =
DescribeDbParameterGroupsRequest.builder()
        .dbParameterGroupName(dbGroupName)
        .maxRecords(20)
        .build();

        DescribeDbParameterGroupsResponse response =
rdsClient.describeDBParameterGroups(groupsRequest);
        List<DBParameterGroup> groups = response.dbParameterGroups();
        for (DBParameterGroup group : groups) {
            System.out.println("The group name is " +
group.dbParameterGroupName());
            System.out.println("The group description is " +
group.description());
        }

    } catch (RdsException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

    public static void createDBParameterGroup(RdsClient rdsClient, String
dbGroupName, String dbParameterGroupFamily) {
        try {
            CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
                .dbParameterGroupName(dbGroupName)
                .dbParameterGroupFamily(dbParameterGroupFamily)
                .description("Created by using the AWS SDK for Java")
                .build();

            CreateDbParameterGroupResponse response =
rdsClient.createDBParameterGroup(groupRequest);
            System.out.println("The group name is " +
response.dbParameterGroup().dbParameterGroupName());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDBEngines(RdsClient rdsClient) {
```

```
try {
    DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
        .defaultOnly(true)
        .engine("mysql")
        .maxRecords(20)
        .build();

    DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
    List<DBEngineVersion> engines = response.dbEngineVersions();

    // Get all DBEngineVersion objects.
    for (DBEngineVersion engineOb : engines) {
        System.out.println("The name of the DB parameter group family for
the database engine is "
            + engineOb.dbParameterGroupFamily());
        System.out.println("The name of the database engine " +
engineOb.engine());
        System.out.println("The version number of the database engine " +
engineOb.engineVersion());
    }

} catch (RdsException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [C reateDBInstance](#)
 - [C reateDBParameter 组](#)
 - [C reateDBSnapshot](#)
 - [D eleteDBInstance](#)
 - [D eleteDBParameter 组](#)
 - [D escribeDBEngine 版本](#)
 - [D escribeDBInstances](#)
 - [D escribeDBParameter 组](#)

- [DescribeDBParameters](#)
- [DescribeDBSnapshots](#)
- [DescribeOrderableDBInstanceOptions](#)
- [ModifyDBParameter 集团](#)

无服务器示例

使用 Lambda 函数连接到亚马逊RDS数据库

以下代码示例说明如何实现连接到数据库的 Lambda 函数。RDS该函数发出一个简单的数据库请求并返回结果。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

使用 Java 在 Lambda 函数中连接到亚马逊RDS数据库。

```
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();
}
}
```

使用SDK适用于 Java 2.x 的 Amazon Redshift 示例

以下代码示例向您展示了如何在 Amazon Redshift 中 AWS SDK for Java 2.x 使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。


每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

开始使用

开始使用 Amazon Redshift

以下代码示例展示了如何开始使用 Amazon Redshift。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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()));
    }
}
```


- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [describeClusters](#)”中的。

主题

- [操作](#)
- [场景](#)

操作

CreateCluster

以下代码示例演示如何使用 CreateCluster。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

创建集群。

```
public static void createCluster(RedshiftClient redshiftClient, String
clusterId, String masterUsername,
                                String masterUserPassword) {
    try {
        CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .masterUsername(masterUsername)
            .masterUserPassword(masterUserPassword)
            .nodeType("ra3.4xlarge")
            .publiclyAccessible(true)
            .numberOfNodes(2)
            .build();

        CreateClusterResponse clusterResponse =
redshiftClient.createCluster(clusterRequest);
        System.out.println("Created cluster " +
clusterResponse.cluster().clusterIdentifier());
    }
}
```

```
    } catch (RedshiftException e) {  
  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateCluster](#)”中的。

CreateTable

以下代码示例演示如何使用 CreateTable。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void createTable(RedshiftDataClient redshiftDataClient, String  
clusterId, String databaseName, String userName) {  
    try {  
        ExecuteStatementRequest createTableRequest =  
ExecuteStatementRequest.builder()  
            .clusterIdentifier(clusterId)  
            .dbUser(userName)  
            .database(databaseName)  
            .sql("CREATE TABLE Movies ("  
                + "id INT PRIMARY KEY, "  
                + "title VARCHAR(100), "  
                + "year INT)")  
            .build();  
  
        redshiftDataClient.executeStatement(createTableRequest);  
        System.out.println("Table created: Movies");  
  
    } catch (RedshiftDataException e) {  
        System.err.println("Error creating table: " + e.getMessage());  
        System.exit(1);  
    }  
}
```

```
    }  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateTable](#)”中的。

DeleteCluster

以下代码示例演示如何使用 DeleteCluster。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

请删除集群。

```
public static void deleteRedshiftCluster(RedshiftClient redshiftClient, String  
clusterId) {  
    try {  
        DeleteClusterRequest deleteClusterRequest =  
DeleteClusterRequest.builder()  
            .clusterIdentifier(clusterId)  
            .skipFinalClusterSnapshot(true)  
            .build();  
  
        DeleteClusterResponse response =  
redshiftClient.deleteCluster(deleteClusterRequest);  
        System.out.println("The status is " +  
response.cluster().clusterStatus());  
  
    } catch (RedshiftException e) {  
        System.err.println(e.getMessage());  
        System.exit(1);  
    }  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteCluster](#)”中的。

DescribeClusters

以下代码示例演示如何使用 DescribeClusters。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

描述集群。

```
public static void waitForClusterReady(RedshiftClient redshiftClient, String
clusterId) {
    boolean clusterReady = false;
    String clusterReadyStr;
    System.out.println("Waiting for cluster to become available. This may take a
few mins.");
    try {
        DescribeClustersRequest clustersRequest =
DescribeClustersRequest.builder()
            .clusterIdentifier(clusterId)
            .build();
        long startTime = System.currentTimeMillis();

        // Loop until the cluster is ready.
        while (!clusterReady) {
            DescribeClustersResponse clusterResponse =
redshiftClient.describeClusters(clustersRequest);
            List<Cluster> clusterList = clusterResponse.clusters();
            for (Cluster cluster : clusterList) {
                clusterReadyStr = cluster.clusterStatus();
                if (clusterReadyStr.contains("available"))
                    clusterReady = true;
                else {
                    long elapsedTimeMillis = System.currentTimeMillis() -
startTime;

                    long elapsedSeconds = elapsedTimeMillis / 1000;
                    long minutes = elapsedSeconds / 60;
                    long seconds = elapsedSeconds % 60;
```

```
        System.out.printf("Elapsed Time: %02d:%02d - Waiting for
cluster... %n", minutes, seconds);
        TimeUnit.SECONDS.sleep(5);
    }
}

long elapsedTimeMillis = System.currentTimeMillis() - startTime;
long elapsedSeconds = elapsedTimeMillis / 1000;
long minutes = elapsedSeconds / 60;
long seconds = elapsedSeconds % 60;

System.out.println(String.format("Cluster is available! Total Elapsed
Time: %02d:%02d", minutes, seconds));

} catch (RedshiftException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeClusters](#)”中的。

DescribeStatement

以下代码示例演示如何使用 DescribeStatement。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void checkStatement(RedshiftDataClient redshiftDataClient, String
sqlId) {
    try {
        DescribeStatementRequest statementRequest =
DescribeStatementRequest.builder()
```

```
        .id(sqlId)
        .build();

    String status;
    while (true) {
        DescribeStatementResponse response =
redshiftDataClient.describeStatement(statementRequest);
        status = response.statusAsString();
        System.out.println("..." + status);

        if (status.compareTo("FAILED") == 0 ) {
            System.out.println("The Query Failed. Ending program");
            System.exit(1);

        } else if (status.compareTo("FINISHED") == 0) {
            break;
        }
        TimeUnit.SECONDS.sleep(1);
    }

    System.out.println("The statement is finished!");

} catch (RedshiftDataException | InterruptedException e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeStatement](#)”中的。

GetStatementResult

以下代码示例演示如何使用 GetStatementResult。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

检查语句结果。

```
public static void getResults(RedshiftDataClient redshiftDataClient, String
statementId) {
    try {
        GetStatementResultRequest resultRequest =
GetStatementResultRequest.builder()
        .id(statementId)
        .build();

        // Extract and print the field values using streams.
        GetStatementResultResponse response =
redshiftDataClient.getStatementResult(resultRequest);
        response.records().stream()
            .flatMap(List::stream)
            .map(Field::stringValue)
            .filter(value -> value != null)
            .forEach(value -> System.out.println("The Movie title field is " +
value));

    } catch (RedshiftDataException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetStatementResult](#)”中的。

Insert

以下代码示例演示如何使用 Insert。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static void popTable(RedshiftDataClient redshiftDataClient, String
clusterId, String databaseName, String userName, String fileName, int number)
throws IOException {
    JsonParser parser = new JsonFactory().createParser(new File(fileName));
    com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
    Iterator<JsonNode> iter = rootNode.iterator();
    ObjectNode currentNode;
    int t = 0;
    while (iter.hasNext()) {
        if (t == number)
            break;
        currentNode = (ObjectNode) iter.next();
        int year = currentNode.get("year").asInt();
        String title = currentNode.get("title").asText();

        // Use SqlParameter to avoid SQL injection.
        List<SqlParameter> parameterList = new ArrayList<>();
        String sqlStatement = "INSERT INTO Movies
VALUES( :id , :title, :year);";

        // Create the parameters.
        SqlParameter idParam = SqlParameter.builder()
            .name("id")
            .value(String.valueOf(t))
            .build();

        SqlParameter titleParam= SqlParameter.builder()
            .name("title")
            .value(title)
            .build();

        SqlParameter yearParam = SqlParameter.builder()
            .name("year")
            .value(String.valueOf(year))
            .build();
        parameterList.add(idParam);
        parameterList.add(titleParam);
        parameterList.add(yearParam);

        try {
            ExecuteStatementRequest insertStatementRequest =
ExecuteStatementRequest.builder()
```



```

        .clusterIdentifier(clusterId)
        .sql(sqlStatement)
        .database(databaseName)
        .dbUser(userName)
        .parameters(parameterList)
        .build();

        redshiftDataClient.executeStatement(insertStatementRequest);
        System.out.println("Inserted: " + title + " (" + year + ")");
        t++;

    } catch (RedshiftDataException e) {
        System.err.println("Error inserting data: " + e.getMessage());
        System.exit(1);
    }
}
System.out.println(t + " records were added to the Movies table. ");
}

```

- 有关API详细信息，请参见[插入AWS SDK for Java 2.x API 参考文献](#)。

ModifyCluster

以下代码示例演示如何使用 ModifyCluster。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

修改集群。

```

public static void modifyCluster(RedshiftClient redshiftClient, String
clusterId) {
    try {
        ModifyClusterRequest modifyClusterRequest =
ModifyClusterRequest.builder()
            .clusterIdentifier(clusterId)

```

```
        .preferredMaintenanceWindow("wed:07:30-wed:08:00")
        .build();

        ModifyClusterResponse clusterResponse =
redshiftClient.modifyCluster(modifyClusterRequest);
        System.out.println("The modified cluster was successfully modified and
has "
            + clusterResponse.cluster().preferredMaintenanceWindow() + " as the
maintenance window");

    } catch (RedshiftException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ModifyCluster](#)”中的。

Query

以下代码示例演示如何使用 Query。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

查询表。

```
public static String queryMoviesByYear(RedshiftDataClient redshiftDataClient,
                                       String database,
                                       String dbUser,
                                       int year,
                                       String clusterId) {

    try {
        String sqlStatement = " SELECT * FROM Movies WHERE year = :year";
        SqlParameter yearParam= SqlParameter.builder()
```

```
        .name("year")
        .value(String.valueOf(year))
        .build();

        ExecuteStatementRequest statementRequest =
ExecuteStatementRequest.builder()
        .clusterIdentifier(clusterId)
        .database(database)
        .dbUser(dbUser)
        .parameters(yearParam)
        .sql(sqlStatement)
        .build();

        ExecuteStatementResponse response =
redshiftDataClient.executeStatement(statementRequest);
        return response.id();

    } catch (RedshiftDataException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}
```

- 有关API详细信息，请参阅AWS SDK for Java 2.x API参考文献中的[查询](#)。

场景

Amazon Redshift 入门

以下代码示例展示了如何使用 Amazon Redshift 表、项目和查询。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import com.fasterxml.jackson.core.JsonFactory;
```

```
import com.fasterxml.jackson.databind.JsonNode;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.node.ObjectNode;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.redshift.RedshiftClient;
import software.amazon.awssdk.services.redshift.model.Cluster;
import software.amazon.awssdk.services.redshift.model.CreateClusterRequest;
import software.amazon.awssdk.services.redshift.model.CreateClusterResponse;
import software.amazon.awssdk.services.redshift.model.DeleteClusterRequest;
import software.amazon.awssdk.services.redshift.model.DeleteClusterResponse;
import software.amazon.awssdk.services.redshift.model.DescribeClustersRequest;
import software.amazon.awssdk.services.redshift.model.DescribeClustersResponse;
import software.amazon.awssdk.services.redshift.model.ModifyClusterRequest;
import software.amazon.awssdk.services.redshift.model.ModifyClusterResponse;
import software.amazon.awssdk.services.redshift.model.RedshiftException;
import software.amazon.awssdk.services.redshiftdata.RedshiftDataClient;
import software.amazon.awssdk.services.redshiftdata.model.DescribeStatementRequest;
import software.amazon.awssdk.services.redshiftdata.model.DescribeStatementResponse;
import software.amazon.awssdk.services.redshiftdata.model.ExecuteStatementRequest;
import software.amazon.awssdk.services.redshiftdata.model.ExecuteStatementResponse;
import software.amazon.awssdk.services.redshiftdata.model.Field;
import software.amazon.awssdk.services.redshiftdata.model.GetStatementResultRequest;
import
    software.amazon.awssdk.services.redshiftdata.model.GetStatementResultResponse;
import software.amazon.awssdk.services.redshiftdata.model.ListDatabasesRequest;
import software.amazon.awssdk.services.redshiftdata.model.RedshiftDataException;
import software.amazon.awssdk.services.redshiftdata.model.SqlParameter;
import
    software.amazon.awssdk.services.redshiftdata.paginators.ListDatabasesIterable;
import com.fasterxml.jackson.core.JsonParser;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.TimeUnit;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 */
```

```

* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
*
This Java example performs these tasks:
*
* 1. Prompts the user for a unique cluster ID or use the default value.
* 2. Creates a Redshift cluster with the specified or default cluster Id value.
* 3. Waits until the Redshift cluster is available for use.
* 4. Lists all databases using a pagination API call.
* 5. Creates a table named "Movies" with fields ID, title, and year.
* 6. Inserts a specified number of records into the "Movies" table by reading the
Movies JSON file.
* 7. Prompts the user for a movie release year.
* 8. Runs a SQL query to retrieve movies released in the specified year.
* 9. Modifies the Redshift cluster.
* 10. Prompts the user for confirmation to delete the Redshift cluster.
* 11. If confirmed, deletes the specified Redshift cluster.
*/

```

```

public class RedshiftScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) throws Exception {
        final String usage = ""

            Usage:
                <jsonFilePath>\s

            Where:
                jsonFilePath - The path to the Movies JSON file (you can locate that
file in ../../../../resources/sample_files/movies.json)
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String jsonFilePath = args[0];
        String userName;
        String userPassword;
        String databaseName = "dev" ;
        Scanner scanner = new Scanner(System.in);

        Region region = Region.US_EAST_1;

```

```
RedshiftClient redshiftClient = RedshiftClient.builder()
    .region(region)
    .build();

RedshiftDataClient redshiftDataClient = RedshiftDataClient.builder()
    .region(region)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the Amazon Redshift SDK Getting Started
scenario.");
System.out.println("""
This Java program demonstrates how to interact with Amazon Redshift by using
the AWS SDK for Java (v2).\s
Amazon Redshift is a fully managed, petabyte-scale data warehouse service
hosted in the cloud.

The program's primary functionalities include cluster creation, verification
of cluster readiness,\s
list databases, table creation, data population within the table, and
execution of SQL statements.
Furthermore, it demonstrates the process of querying data from the Movie
table.\s

Upon completion of the program, all AWS resources are cleaned up.
""");

System.out.println("Lets get started...");
System.out.println("Please enter your user name (default is awsuser)");
String user = scanner.nextLine();
userName = user.isEmpty() ? "awsuser" : user;
System.out.println(DASHES);
System.out.println("Please enter your user password (default is
AwsUser1000)");
String userpass = scanner.nextLine();
userPassword = userpass.isEmpty() ? "AwsUser1000" : userpass;
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("A Redshift cluster refers to the collection of computing
resources and storage that work together to process and analyze large volumes of
data.");
System.out.println("Enter a cluster id value (default is redshift-cluster-
movies): ");
```

```
String userClusterId = scanner.nextLine();
String clusterId = userClusterId.isEmpty() ? "redshift-cluster-movies" :
userClusterId;
createCluster(redshiftClient, clusterId, userName, userPassword);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Wait until "+clusterId+" is available.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
waitForClusterReady(redshiftClient, clusterId);
System.out.println(DASHES);

System.out.println(DASHES);
String databaseInfo = ""
    When you created $clusteridD, the dev database is created by default and
used in this scenario.\s

    To create a custom database, you need to have a CREATEDB privilege.\s
    For more information, see the documentation here: https://
docs.aws.amazon.com/redshift/latest/dg/r\_CREATE\_DATABASE.html.
"".replace("$clusteridD", clusterId);

System.out.println(databaseInfo);
System.out.print("Press Enter to continue...");
scanner.nextLine();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("List databases in "+clusterId);
System.out.print("Press Enter to continue...");
scanner.nextLine();
listAllDatabases(redshiftDataClient, clusterId, userName, databaseName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Now you will create a table named Movies.");
System.out.print("Press Enter to continue...");
scanner.nextLine();
createTable(redshiftDataClient, clusterId, databaseName, userName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Populate the Movies table using the Movies.json file.");
```

```
        System.out.println("Specify the number of records you would like to add to
the Movies Table.");
        System.out.println("Please enter a value between 50 and 200.");
        int numRecords;
        do {
            System.out.print("Enter a value: ");
            while (!scanner.hasNextInt()) {
                System.out.println("Invalid input. Please enter a value between 50
and 200.");
                System.out.print("Enter a year: ");
                scanner.next();
            }
            numRecords = scanner.nextInt();
        } while (numRecords < 50 || numRecords > 200);
        popTable(redshiftDataClient, clusterId, databaseName, userName,
jsonFilePath, numRecords);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Query the Movies table by year. Enter a value between
2012-2014.");
        int movieYear;
        do {
            System.out.print("Enter a year: ");
            while (!scanner.hasNextInt()) {
                System.out.println("Invalid input. Please enter a valid year between
2012 and 2014.");
                System.out.print("Enter a year: ");
                scanner.next();
            }
            movieYear = scanner.nextInt();
            scanner.nextLine();
        } while (movieYear < 2012 || movieYear > 2014);

        String id = queryMoviesByYear(redshiftDataClient, databaseName, userName,
movieYear, clusterId);
        System.out.println("The identifier of the statement is " + id);
        checkStatement(redshiftDataClient, id);
        getResults(redshiftDataClient, id);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Now you will modify the Redshift cluster.");
        System.out.print("Press Enter to continue...");
```



```
scanner.nextLine();
modifyCluster(redshiftClient, clusterId);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Would you like to delete the Amazon Redshift cluster?
(y/n)");
String delAns = scanner.nextLine().trim();
if (delAns.equalsIgnoreCase("y")) {
    System.out.println("You selected to delete " +clusterId);
    System.out.print("Press Enter to continue...");
    scanner.nextLine();
    deleteRedshiftCluster(redshiftClient, clusterId);
} else {
    System.out.println("The "+clusterId +" was not deleted");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("This concludes the Amazon Redshift SDK Getting Started
scenario.");
System.out.println(DASHES);
}

public static void listAllDatabases(RedshiftDataClient redshiftDataClient,
String clusterId, String dbUser, String database) {
    try {
        ListDatabasesRequest databasesRequest = ListDatabasesRequest.builder()
            .clusterIdentifier(clusterId)
            .dbUser(dbUser)
            .database(database)
            .build();

        ListDatabasesIterable listDatabasesIterable =
redshiftDataClient.listDatabasesPaginator(databasesRequest);
        listDatabasesIterable.stream()
            .flatMap(r -> r.databases().stream())
            .forEach(db -> System.out
                .println("The database name is : " + db));

    } catch (RedshiftDataException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
}

    public static void deleteRedshiftCluster(RedshiftClient redshiftClient, String
clusterId) {
        try {
            DeleteClusterRequest deleteClusterRequest =
DeleteClusterRequest.builder()
                .clusterIdentifier(clusterId)
                .skipFinalClusterSnapshot(true)
                .build();

            DeleteClusterResponse response =
redshiftClient.deleteCluster(deleteClusterRequest);
            System.out.println("The status is " +
response.cluster().clusterStatus());

        } catch (RedshiftException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void popTable(RedshiftDataClient redshiftDataClient, String
clusterId, String databaseName, String userName, String fileName, int number)
throws IOException {
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0;
        while (iter.hasNext()) {
            if (t == number)
                break;
            currentNode = (ObjectNode) iter.next();
            int year = currentNode.get("year").asInt();
            String title = currentNode.get("title").asText();

            // Use SqlParameter to avoid SQL injection.
            List<SqlParameter> parameterList = new ArrayList<>();
            String sqlStatement = "INSERT INTO Movies
VALUES( :id , :title, :year);";

            // Create the parameters.
```

```
        SqlParameter idParam = SqlParameter.builder()
            .name("id")
            .value(String.valueOf(t))
            .build();

        SqlParameter titleParam= SqlParameter.builder()
            .name("title")
            .value(title)
            .build();

        SqlParameter yearParam = SqlParameter.builder()
            .name("year")
            .value(String.valueOf(year))
            .build();
        parameterList.add(idParam);
        parameterList.add(titleParam);
        parameterList.add(yearParam);

        try {
            ExecuteStatementRequest insertStatementRequest =
ExecuteStatementRequest.builder()
                .clusterIdentifier(clusterId)
                .sql(sqlStatement)
                .database(databaseName)
                .dbUser(userName)
                .parameters(parameterList)
                .build();

            redshiftDataClient.executeStatement(insertStatementRequest);
            System.out.println("Inserted: " + title + " (" + year + ")");
            t++;

        } catch (RedshiftDataException e) {
            System.err.println("Error inserting data: " + e.getMessage());
            System.exit(1);
        }
    }
    System.out.println(t + " records were added to the Movies table. ");
}

public static void checkStatement(RedshiftDataClient redshiftDataClient, String
sqlId) {
    try {
```

```
        DescribeStatementRequest statementRequest =
DescribeStatementRequest.builder()
        .id(sqlId)
        .build();

        String status;
        while (true) {
            DescribeStatementResponse response =
redshiftDataClient.describeStatement(statementRequest);
            status = response.statusAsString();
            System.out.println("..." + status);

            if (status.compareTo("FAILED") == 0 ) {
                System.out.println("The Query Failed. Ending program");
                System.exit(1);
            } else if (status.compareTo("FINISHED") == 0) {
                break;
            }
            TimeUnit.SECONDS.sleep(1);
        }

        System.out.println("The statement is finished!");

    } catch (RedshiftDataException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void modifyCluster(RedshiftClient redshiftClient, String
clusterId) {
    try {
        ModifyClusterRequest modifyClusterRequest =
ModifyClusterRequest.builder()
        .clusterIdentifier(clusterId)
        .preferredMaintenanceWindow("wed:07:30-wed:08:00")
        .build();

        ModifyClusterResponse clusterResponse =
redshiftClient.modifyCluster(modifyClusterRequest);
        System.out.println("The modified cluster was successfully modified and
has "
```

```
        + clusterResponse.cluster().preferredMaintenanceWindow() + " as the
maintenance window");

    } catch (RedshiftException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static String queryMoviesByYear(RedshiftDataClient redshiftDataClient,
                                       String database,
                                       String dbUser,
                                       int year,
                                       String clusterId) {

    try {
        String sqlStatement = " SELECT * FROM Movies WHERE year = :year";
        SqlParameter yearParam= SqlParameter.builder()
            .name("year")
            .value(String.valueOf(year))
            .build();

        ExecuteStatementRequest statementRequest =
ExecuteStatementRequest.builder()
            .clusterIdentifier(clusterId)
            .database(database)
            .dbUser(dbUser)
            .parameters(yearParam)
            .sql(sqlStatement)
            .build();

        ExecuteStatementResponse response =
redshiftDataClient.executeStatement(statementRequest);
        return response.id();

    } catch (RedshiftDataException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
}

public static void getResults(RedshiftDataClient redshiftDataClient, String
statementId) {
```

```
        try {
            GetStatementResultRequest resultRequest =
GetStatementResultRequest.builder()
                .id(statementId)
                .build();

            // Extract and print the field values using streams.
            GetStatementResultResponse response =
redshiftDataClient.getStatementResult(resultRequest);
            response.records().stream()
                .flatMap(List::stream)
                .map(Field::stringValue)
                .filter(value -> value != null)
                .forEach(value -> System.out.println("The Movie title field is " +
value));

        } catch (RedshiftDataException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void waitForClusterReady(RedshiftClient redshiftClient, String
clusterId) {
        boolean clusterReady = false;
        String clusterReadyStr;
        System.out.println("Waiting for cluster to become available. This may take a
few mins.");
        try {
            DescribeClustersRequest clustersRequest =
DescribeClustersRequest.builder()
                .clusterIdentifier(clusterId)
                .build();
            long startTime = System.currentTimeMillis();

            // Loop until the cluster is ready.
            while (!clusterReady) {
                DescribeClustersResponse clusterResponse =
redshiftClient.describeClusters(clustersRequest);
                List<Cluster> clusterList = clusterResponse.clusters();
                for (Cluster cluster : clusterList) {
                    clusterReadyStr = cluster.clusterStatus();
                    if (clusterReadyStr.contains("available"))
                        clusterReady = true;
                }
            }
        }
    }
}
```

```
        else {
            long elapsedTimeMillis = System.currentTimeMillis() -
startTime;

            long elapsedSeconds = elapsedTimeMillis / 1000;
            long minutes = elapsedSeconds / 60;
            long seconds = elapsedSeconds % 60;

            System.out.printf("Elapsed Time: %02d:%02d - Waiting for
cluster... %n", minutes, seconds);
            TimeUnit.SECONDS.sleep(5);
        }
    }

    long elapsedTimeMillis = System.currentTimeMillis() - startTime;
    long elapsedSeconds = elapsedTimeMillis / 1000;
    long minutes = elapsedSeconds / 60;
    long seconds = elapsedSeconds % 60;

    System.out.println(String.format("Cluster is available! Total Elapsed
Time: %02d:%02d", minutes, seconds));

    } catch (RedshiftException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void createTable(RedshiftDataClient redshiftDataClient, String
clusterId, String databaseName, String userName) {
    try {
        ExecuteStatementRequest createTableRequest =
ExecuteStatementRequest.builder()
            .clusterIdentifier(clusterId)
            .dbUser(userName)
            .database(databaseName)
            .sql("CREATE TABLE Movies ("
                + "id INT PRIMARY KEY, "
                + "title VARCHAR(100), "
                + "year INT)")
            .build();

        redshiftDataClient.executeStatement(createTableRequest);
        System.out.println("Table created: Movies");
    }
}
```

```
        } catch (RedshiftDataException e) {
            System.err.println("Error creating table: " + e.getMessage());
            System.exit(1);
        }
    }

    public static void createCluster(RedshiftClient redshiftClient, String
clusterId, String masterUsername,
                                   String masterUserPassword) {
        try {
            CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
                .clusterIdentifier(clusterId)
                .masterUsername(masterUsername)
                .masterUserPassword(masterUserPassword)
                .nodeType("ra3.4xlarge")
                .publiclyAccessible(true)
                .numberOfNodes(2)
                .build();

            CreateClusterResponse clusterResponse =
redshiftClient.createCluster(clusterRequest);
            System.out.println("Created cluster " +
clusterResponse.cluster().clusterIdentifier());

        } catch (RedshiftException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。

- [createCluster](#)
- [describeClusters](#)
- [describeStatement](#)
- [executeStatement](#)
- [getStatementResult](#)
- [listDatabasesPaginator](#)

- [modifyCluster](#)

使用适用于 Java 2.x 的亚马逊 Rekogn SDK ition 示例

以下代码示例向您展示了如何使用 AWS SDK for Java 2.x 与 Amazon Rekognition 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)
- [场景](#)

操作

CompareFaces

以下代码示例演示如何使用 CompareFaces。

有关更多信息，请参阅[比较图像中的人脸](#)。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.CompareFacesRequest;
```

```
import software.amazon.awssdk.services.rekognition.model.CompareFacesResponse;
import software.amazon.awssdk.services.rekognition.model.CompareFacesMatch;
import software.amazon.awssdk.services.rekognition.model.ComparedFace;
import software.amazon.awssdk.services.rekognition.model.BoundingBox;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CompareFaces {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <pathSource> <pathTarget>

                Where:
                pathSource - The path to the source image (for example, C:\\AWS\\
\\pic1.png).\s
                pathTarget - The path to the target image (for example, C:\\AWS\\
\\pic2.png).\s

                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        Float similarityThreshold = 70F;
        String sourceImage = args[0];
        String targetImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();
    }
}
```

```
        compareTwoFaces(rekClient, similarityThreshold, sourceImage, targetImage);
        rekClient.close();
    }

    public static void compareTwoFaces(RecognitionClient rekClient, Float
similarityThreshold, String sourceImage,
        String targetImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            InputStream tarStream = new FileInputStream(targetImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            SdkBytes targetBytes = SdkBytes.fromInputStream(tarStream);

            // Create an Image object for the source image.
            Image souImage = Image.builder()
                .bytes(sourceBytes)
                .build();

            Image tarImage = Image.builder()
                .bytes(targetBytes)
                .build();

            CompareFacesRequest facesRequest = CompareFacesRequest.builder()
                .sourceImage(souImage)
                .targetImage(tarImage)
                .similarityThreshold(similarityThreshold)
                .build();

            // Compare the two images.
            CompareFacesResponse compareFacesResult =
rekClient.compareFaces(facesRequest);
            List<CompareFacesMatch> faceDetails = compareFacesResult.faceMatches();
            for (CompareFacesMatch match : faceDetails) {
                ComparedFace face = match.face();
                BoundingBox position = face.boundingBox();
                System.out.println("Face at " + position.left().toString()
                    + " " + position.top()
                    + " matches with " + face.confidence().toString()
                    + "% confidence.");
            }
            List<ComparedFace> uncompered = compareFacesResult.unmatchedFaces();
            System.out.println("There was " + uncompered.size() + " face(s) that did
not match");
        }
    }
}
```

```
        System.out.println("Source image rotation: " +
compareFacesResult.sourceImageOrientationCorrection());
        System.out.println("target image rotation: " +
compareFacesResult.targetImageOrientationCorrection());

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println("Failed to load source image " + sourceImage);
        System.exit(1);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CompareFaces](#)”中的。

CreateCollection

以下代码示例演示如何使用 CreateCollection。

有关更多信息，请参阅[创建集合](#)。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.CreateCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
*/
public class CreateCollection {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <collectionName>\s

            Where:
                collectionName - The name of the collection.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        System.out.println("Creating collection: " + collectionId);
        createMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    public static void createMyCollection(RekognitionClient rekClient, String
collectionId) {
        try {
            CreateCollectionRequest collectionRequest =
CreateCollectionRequest.builder()
                .collectionId(collectionId)
                .build();

            CreateCollectionResponse collectionResponse =
rekClient.createCollection(collectionRequest);
            System.out.println("CollectionArn: " +
collectionResponse.collectionArn());
            System.out.println("Status code: " +
collectionResponse.statusCode().toString());

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

```
        System.exit(1);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateCollection](#)”中的。

DeleteCollection

以下代码示例演示如何使用 DeleteCollection。

有关更多信息，请参阅[删除集合](#)。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.DeleteCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionId>\s
```

```
        Where:
            collectionId - The id of the collection to delete.\s
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String collectionId = args[0];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    System.out.println("Deleting collection: " + collectionId);
    deleteMyCollection(rekClient, collectionId);
    rekClient.close();
}

public static void deleteMyCollection(RekognitionClient rekClient, String
collectionId) {
    try {
        DeleteCollectionRequest deleteCollectionRequest =
DeleteCollectionRequest.builder()
            .collectionId(collectionId)
            .build();

        DeleteCollectionResponse deleteCollectionResponse =
rekClient.deleteCollection(deleteCollectionRequest);
        System.out.println(collectionId + ": " +
deleteCollectionResponse.statusCode().toString());

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteCollection](#)”中的。

DeleteFaces

以下代码示例演示如何使用 DeleteFaces。

有关更多信息，请参阅[从集中删除人脸](#)。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteFacesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteFacesFromCollection {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <collectionId> <faceId>\s

            Where:
                collectionId - The id of the collection from which faces are
deleted.\s

                faceId - The id of the face to delete.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
        }
    }
}
```



```
        System.exit(1);
    }

    String collectionId = args[0];
    String faceId = args[1];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    System.out.println("Deleting collection: " + collectionId);
    deleteFacesCollection(rekClient, collectionId, faceId);
    rekClient.close();
}

public static void deleteFacesCollection(RekognitionClient rekClient,
    String collectionId,
    String faceId) {

    try {
        DeleteFacesRequest deleteFacesRequest = DeleteFacesRequest.builder()
            .collectionId(collectionId)
            .faceIds(faceId)
            .build();

        rekClient.deleteFaces(deleteFacesRequest);
        System.out.println("The face was deleted from the collection.");

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteFaces](#)”中的。

DescribeCollection

以下代码示例演示如何使用 DescribeCollection。

有关更多信息，请参阅[描述集合](#)。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DescribeCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.DescribeCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeCollection {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <collectionName>

            Where:
                collectionName - The name of the Amazon Rekognition collection.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionName = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();
```

```
        describeColl(rekClient, collectionName);
        rekClient.close();
    }

    public static void describeColl(RekognitionClient rekClient, String
collectionName) {
        try {
            DescribeCollectionRequest describeCollectionRequest =
DescribeCollectionRequest.builder()
                .collectionId(collectionName)
                .build();

            DescribeCollectionResponse describeCollectionResponse = rekClient
                .describeCollection(describeCollectionRequest);

            System.out.println("Collection Arn : " +
describeCollectionResponse.collectionARN());
            System.out.println("Created : " +
describeCollectionResponse.creationTimestamp().toString());

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeCollection](#)”中的。

DetectFaces

以下代码示例演示如何使用 DetectFaces。

有关更多信息，请参阅[检测图像中的人脸](#)。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.DetectFacesRequest;
import software.amazon.awssdk.services.rekognition.model.DetectFacesResponse;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.Attribute;
import software.amazon.awssdk.services.rekognition.model.FaceDetail;
import software.amazon.awssdk.services.rekognition.model.AgeRange;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectFaces {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <sourceImage>

                Where:
                    sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
```

```
        .build();

        detectFacesinImage(rekClient, sourceImage);
        rekClient.close();
    }

    public static void detectFacesinImage(RekognitionClient rekClient, String
sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            // Create an Image object for the source image.
            Image souImage = Image.builder()
                .bytes(sourceBytes)
                .build();

            DetectFacesRequest facesRequest = DetectFacesRequest.builder()
                .attributes(Attribute.ALL)
                .image(souImage)
                .build();

            DetectFacesResponse facesResponse = rekClient.detectFaces(facesRequest);
            List<FaceDetail> faceDetails = facesResponse.faceDetails();
            for (FaceDetail face : faceDetails) {
                AgeRange ageRange = face.ageRange();
                System.out.println("The detected face is estimated to be between "
                    + ageRange.low().toString() + " and " +
ageRange.high().toString()
                    + " years old.");

                System.out.println("There is a smile : " +
face.smile().value().toString());
            }

        } catch (RekognitionException | FileNotFoundException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DetectFaces](#)”中的。

DetectLabels

以下代码示例演示如何使用 DetectLabels。

有关更多信息，请参阅[检测图像中的标签](#)。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.DetectLabelsRequest;
import software.amazon.awssdk.services.rekognition.model.DetectLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.Label;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectLabels {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <sourceImage>

                Where:
```

```
        sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s
        """";

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String sourceImage = args[0];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    detectImageLabels(rekClient, sourceImage);
    rekClient.close();
}

public static void detectImageLabels(RekognitionClient rekClient, String
sourceImage) {
    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        // Create an Image object for the source image.
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        DetectLabelsRequest detectLabelsRequest = DetectLabelsRequest.builder()
            .image(souImage)
            .maxLabels(10)
            .build();

        DetectLabelsResponse labelsResponse =
rekClient.detectLabels(detectLabelsRequest);
        List<Label> labels = labelsResponse.labels();
        System.out.println("Detected labels for the given photo");
        for (Label label : labels) {
            System.out.println(label.name() + ": " +
label.confidence().toString());
        }
    }
}
```

```
        } catch (RekognitionException | FileNotFoundException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DetectLabels](#)”中的。

DetectModerationLabels

以下代码示例演示如何使用 DetectModerationLabels。

有关更多信息，请参阅[检测不适宜的图像](#)。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import
    software.amazon.awssdk.services.rekognition.model.DetectModerationLabelsRequest;
import
    software.amazon.awssdk.services.rekognition.model.DetectModerationLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.ModerationLabel;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```



```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class DetectModerationLabels {

    public static void main(String[] args) {
        final String usage = ""

            Usage:    <sourceImage>

            Where:
                sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s
                """;

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        detectModLabels(rekClient, sourceImage);
        rekClient.close();
    }

    public static void detectModLabels(RekognitionClient rekClient, String
sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(sourceImage);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                .bytes(sourceBytes)
                .build();

            DetectModerationLabelsRequest moderationLabelsRequest =
DetectModerationLabelsRequest.builder()
                .image(souImage)
```

```
        .minConfidence(60F)
        .build();

        DetectModerationLabelsResponse moderationLabelsResponse = rekClient
            .detectModerationLabels(moderationLabelsRequest);
        List<ModerationLabel> labels =
            moderationLabelsResponse.moderationLabels();
        System.out.println("Detected labels for image");
        for (ModerationLabel label : labels) {
            System.out.println("Label: " + label.name()
                + "\n Confidence: " + label.confidence().toString() + "%"
                + "\n Parent:" + label.parentName());
        }
    } catch (RekognitionException | FileNotFoundException e) {
        e.printStackTrace();
        System.exit(1);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DetectModerationLabels](#)”中的。

DetectText

以下代码示例演示如何使用 DetectText。

有关更多信息，请参阅[检测图像中的文本](#)。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DetectTextRequest;
```

```
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.DetectTextResponse;
import software.amazon.awssdk.services.rekognition.model.TextDetection;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectText {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <sourceImage>

            Where:
                sourceImage - The path to the image that contains text (for
example, C:\\AWS\\pic1.png).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String sourceImage = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        detectTextLabels(rekClient, sourceImage);
        rekClient.close();
    }
}
```

```
public static void detectTextLabels(RekognitionClient rekClient, String
sourceImage) {
    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        DetectTextRequest textRequest = DetectTextRequest.builder()
            .image(souImage)
            .build();

        DetectTextResponse textResponse = rekClient.detectText(textRequest);
        List<TextDetection> textCollection = textResponse.textDetections();
        System.out.println("Detected lines and words");
        for (TextDetection text : textCollection) {
            System.out.println("Detected: " + text.detectedText());
            System.out.println("Confidence: " + text.confidence().toString());
            System.out.println("Id : " + text.id());
            System.out.println("Parent Id: " + text.parentId());
            System.out.println("Type: " + text.type());
            System.out.println();
        }

    } catch (RekognitionException | FileNotFoundException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DetectText](#)”中的。

IndexFaces

以下代码示例演示如何使用 IndexFaces。

有关更多信息，请参阅[将人脸添加到集合中](#)。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.IndexFacesResponse;
import software.amazon.awssdk.services.rekognition.model.IndexFacesRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.QualityFilter;
import software.amazon.awssdk.services.rekognition.model.Attribute;
import software.amazon.awssdk.services.rekognition.model.FaceRecord;
import software.amazon.awssdk.services.rekognition.model.UnindexedFace;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Reason;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AddFacesToCollection {
    public static void main(String[] args) {

        final String usage = ""

            Usage:      <collectionId> <sourceImage>

            Where:
                collectionName - The name of the collection.
```

```
        sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s
        """";

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String collectionId = args[0];
    String sourceImage = args[1];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    addToCollection(rekClient, collectionId, sourceImage);
    rekClient.close();
}

public static void addToCollection(RekognitionClient rekClient, String
collectionId, String sourceImage) {
    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        IndexFacesRequest facesRequest = IndexFacesRequest.builder()
            .collectionId(collectionId)
            .image(souImage)
            .maxFaces(1)
            .qualityFilter(QualityFilter.AUTO)
            .detectionAttributes(Attribute.DEFAULT)
            .build();

        IndexFacesResponse facesResponse = rekClient.indexFaces(facesRequest);
        System.out.println("Results for the image");
        System.out.println("\n Faces indexed:");
        List<FaceRecord> faceRecords = facesResponse.faceRecords();
        for (FaceRecord faceRecord : faceRecords) {
            System.out.println("  Face ID: " + faceRecord.face().faceId());
        }
    }
}
```

```
        System.out.println(" Location:" +
faceRecord.faceDetail().boundingBox().toString());
    }

    List<UnindexedFace> unindexedFaces = facesResponse.unindexedFaces();
    System.out.println("Faces not indexed:");
    for (UnindexedFace unindexedFace : unindexedFaces) {
        System.out.println(" Location:" +
unindexedFace.faceDetail().boundingBox().toString());
        System.out.println(" Reasons:");
        for (Reason reason : unindexedFace.reasons()) {
            System.out.println("Reason: " + reason);
        }
    }

} catch (RekognitionException | FileNotFoundException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [IndexFaces](#)”中的。

ListCollections

以下代码示例演示如何使用 ListCollections。

有关更多信息，请参阅[列出集合](#)。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```


- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListCollections](#)”中的。

ListFaces

以下代码示例演示如何使用 ListFaces。

有关更多信息，请参阅[列出集合中的人脸](#)。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListFaces](#)”中的。

RecognizeCelebrities

以下代码示例演示如何使用 RecognizeCelebrities。

有关更多信息，请参阅[识别图像中的名人](#)。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;
import
    software.amazon.awssdk.services.rekognition.model.RecognizeCelebritiesRequest;
import
    software.amazon.awssdk.services.rekognition.model.RecognizeCelebritiesResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.Celebrity;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RecognizeCelebrities {
    public static void main(String[] args) {
        final String usage = ""
            Usage:    <sourceImage>

            Where:
```

```
        sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String sourceImage = args[0];
    Region region = Region.US_EAST_1;
    RekognitionClient rekClient = RekognitionClient.builder()
        .region(region)
        .build();

    System.out.println("Locating celebrities in " + sourceImage);
    recognizeAllCelebrities(rekClient, sourceImage);
    rekClient.close();
}

public static void recognizeAllCelebrities(RekognitionClient rekClient, String
sourceImage) {
    try {
        InputStream sourceStream = new FileInputStream(sourceImage);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
        Image souImage = Image.builder()
            .bytes(sourceBytes)
            .build();

        RecognizeCelebritiesRequest request =
        RecognizeCelebritiesRequest.builder()
            .image(souImage)
            .build();

        RecognizeCelebritiesResponse result =
        rekClient.recognizeCelebrities(request);
        List<Celebrity> celebs = result.celebrityFaces();
        System.out.println(celebs.size() + " celebrity(s) were recognized.\n");
        for (Celebrity celebrity : celebs) {
            System.out.println("Celebrity recognized: " + celebrity.name());
            System.out.println("Celebrity ID: " + celebrity.id());

            System.out.println("Further information (if available):");
            for (String url : celebrity.urls()) {
```

```
        System.out.println(url);
    }
    System.out.println();
}
System.out.println(result.unrecognizedFaces().size() + " face(s) were
unrecognized.");

} catch (RekognitionException | FileNotFoundException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [RecognizeCelebrities](#)”中的。

SearchFaces

以下代码示例演示如何使用 SearchFaces。

有关更多信息，请参阅[搜索人脸 \(面容 ID\)](#)。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.SearchFacesByImageRequest;
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.SearchFacesByImageResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
```

```
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SearchFaceMatchingImageCollection {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <collectionId> <sourceImage>

            Where:
                collectionId - The id of the collection. \s
                sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\\s

            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String sourceImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
            .region(region)
            .build();

        System.out.println("Searching for a face in a collections");
        searchFaceInCollection(rekClient, collectionId, sourceImage);
        rekClient.close();
    }

    public static void searchFaceInCollection(RekognitionClient rekClient, String
collectionId, String sourceImage) {
        try {
```

```
InputStream sourceStream = new FileInputStream(new File(sourceImage));
SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
Image souImage = Image.builder()
    .bytes(sourceBytes)
    .build();

SearchFacesByImageRequest facesByImageRequest =
SearchFacesByImageRequest.builder()
    .image(souImage)
    .maxFaces(10)
    .faceMatchThreshold(70F)
    .collectionId(collectionId)
    .build();

SearchFacesByImageResponse imageResponse =
rekClient.searchFacesByImage(facesByImageRequest);
System.out.println("Faces matching in the collection");
List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
for (FaceMatch face : faceImageMatches) {
    System.out.println("The similarity level is " + face.similarity());
    System.out.println();
}

} catch (RekognitionException | FileNotFoundException e) {
    System.out.println(e.getMessage());
    System.exit(1);
}
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [SearchFaces](#)”中的。

SearchFacesByImage

以下代码示例演示如何使用 SearchFacesByImage。

有关更多信息，请参阅[搜索人脸 \(图像\)](#)。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.SearchFacesRequest;
import software.amazon.awssdk.services.rekognition.model.SearchFacesResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SearchFaceMatchingIdCollection {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <collectionId> <sourceImage>

                Where:
                    collectionId - The id of the collection. \s
                    sourceImage - The path to the image (for example, C:\\AWS\\
\\pic1.png).\s

                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
```



```
String faceId = args[1];
Region region = Region.US_EAST_1;
RekognitionClient rekClient = RekognitionClient.builder()
    .region(region)
    .build();

System.out.println("Searching for a face in a collections");
searchFaceById(rekClient, collectionId, faceId);
rekClient.close();
}

public static void searchFaceById(RekognitionClient rekClient, String
collectionId, String faceId) {
    try {
        SearchFacesRequest searchFacesRequest = SearchFacesRequest.builder()
            .collectionId(collectionId)
            .faceId(faceId)
            .faceMatchThreshold(70F)
            .maxFaces(2)
            .build();

        SearchFacesResponse imageResponse =
rekClient.searchFaces(searchFacesRequest);
        System.out.println("Faces matching in the collection");
        List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
        for (FaceMatch face : faceImageMatches) {
            System.out.println("The similarity level is " + face.similarity());
            System.out.println();
        }

    } catch (RekognitionException e) {
        System.out.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [SearchFacesByImage](#)”中的。

场景

检测视频中的信息

以下代码示例展示了如何：

- 启动 Amazon Rekognition 任务，检测视频中的人物、对象和文本等元素。
- 查看任务状态，直到任务完成。
- 输出每个任务检测到的元素列表。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

从位于 Amazon S3 存储桶中的视频获取名人结果。

```
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 vidObj = Video.builder()
            .s3Object(s3Obj)
            .build();

        StartCelebrityRecognitionRequest recognitionRequest =
StartCelebrityRecognitionRequest.builder()
            .jobTag("Celebrities")
            .notificationChannel(channel)
            .video(vidObj)
            .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);
    }
}
}
}

```

通过标签检测操作检测视频中的标签。

```

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 vidObj = Video.builder()
            .s3Object(s3Obj)
            .build();

        StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
        .jobTag("DetectingLabels")
        .notificationChannel(channel)
        .video(vidObj)
        .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(RecognitionClient 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());
            }
        } while (labelDetectionResult.nextToken() != null);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```
        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);
        }
    }
}
```

检测存储在 Amazon S3 存储桶内的视频中的人脸

```
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 vid0b = Video.builder()
        .s3object(s3obj)
        .build();

    StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
    .jobTag("DetectingLabels")
    .notificationChannel(channel)
    .video(vid0b)
    .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);
    }
}
}
```

检测存储在 Amazon S3 存储桶内的视频中的不当或冒犯性内容。

```
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 vidObj = Video.builder()
            .s3Object(s3obj)
            .build();

        StartContentModerationRequest modDetectionRequest =
StartContentModerationRequest.builder()
            .jobTag("Moderation")
            .notificationChannel(channel)
            .video(vidObj)
            .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);
        }
    }
}

```

检测存储在 Amazon S3 存储桶内的视频中的技术提示片段和镜头检测片段。

```

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(RecognitionClient rekClient,
        NotificationChannel channel,
        String bucket,
        String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                .bucket(bucket)
                .name(video)
                .build();

            Video vidObj = 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(vidObj)
                .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 (RecognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}

```

检测存储在 Amazon S3 存储桶内的视频中的文本。

```

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.RecognitionException;
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 vid0b = Video.builder()
        .s3object(s3obj)
        .build();

    StartTextDetectionRequest labelDetectionRequest =
StartTextDetectionRequest.builder()
        .jobTag("DetectingLabels")
        .notificationChannel(channel)
        .video(vid0b)
        .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);
    }
}
}
```

检测存储在 Amazon S3 存储桶内的视频中的人物。

```
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 vidObj = Video.builder()
            .s3Object(s3obj)
```

```
        .build());

        StartPersonTrackingRequest personTrackingRequest =
StartPersonTrackingRequest.builder()
        .jobTag("DetectingLabels")
        .video(vid0b)
        .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);
    }
}
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。

- [GetCelebrityRecognition](#)
- [GetContentModeration](#)
- [GetLabelDetection](#)
- [GetPersonTracking](#)
- [GetSegmentDetection](#)
- [GetTextDetection](#)
- [StartCelebrityRecognition](#)
- [StartContentModeration](#)
- [StartLabelDetection](#)
- [StartPersonTracking](#)
- [StartSegmentDetection](#)
- [StartTextDetection](#)

使用SDK适用于 Java 2.x 的 Route 53 域名注册示例

以下代码示例向您展示了如何在 Route 53 域注册中 AWS SDK for Java 2.x 使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

开始使用

开始使用 Route 53 域注册

以下代码示例显示如何开始使用 Route 53 域注册。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);  
    }  
}  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListPrices](#)”中的。

主题

- [操作](#)
- [场景](#)

操作

CheckDomainAvailability

以下代码示例演示如何使用 CheckDomainAvailability。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CheckDomainAvailability](#)”中的。

CheckDomainTransferability

以下代码示例演示如何使用 CheckDomainTransferability。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```



```
    }  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CheckDomainTransferability](#)”中的。

GetDomainDetail

以下代码示例演示如何使用 GetDomainDetail。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);  
    }  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetDomainDetail](#)”中的。

GetDomainSuggestions

以下代码示例演示如何使用 `GetDomainSuggestions`。

SDK适用于 Java 2.x

Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetDomainSuggestions](#)”中的。

GetOperationDetail

以下代码示例演示如何使用 GetOperationDetail。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetOperationDetail](#)”中的。

ListDomains

以下代码示例演示如何使用 ListDomains。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListDomains](#)”中的。

ListOperations

以下代码示例演示如何使用 ListOperations。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListOperations](#)”中的。

ListPrices

以下代码示例演示如何使用 ListPrices。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListPrices](#)”中的。

RegisterDomain

以下代码示例演示如何使用 RegisterDomain。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";  
    }
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [RegisterDomain](#)”中的。

ViewBilling

以下代码示例演示如何使用 ViewBilling。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);  
    }  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ViewBilling](#)”中的。

场景

域入门

以下代码示例演示了操作流程：

- 列出当前域，并列过去一年的操作。
- 查看过去一年的账单，并查看域类型的价格。
- 获取域建议。
- 检查域可用性和可转移性。
- (可选) 申请域注册。
- 获取操作详细信息。
- (可选) 获取域详细信息。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including 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 xxxxxxxx not found in xxxxxxxx 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);
        }
    }
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [CheckDomainAvailability](#)
 - [CheckDomainTransferability](#)
 - [GetDomainDetail](#)
 - [GetDomainSuggestions](#)
 - [GetOperationDetail](#)
 - [ListDomains](#)
 - [ListOperations](#)
 - [ListPrices](#)
 - [RegisterDomain](#)
 - [ViewBilling](#)

SDK适用于 Java 2.x 的 Amazon S3 示例

以下代码示例向您展示了如何在 Amazon S3 中使用来执行操作和实现常见场景。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

开始使用

Hello Amazon S3

以下代码示例展示了如何开始使用 Amazon S3。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
*/
public class HelloS3 {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBuckets(s3);
    }

    public static void listBuckets(S3Client s3) {
        try {
            ListBucketsResponse response = s3.listBuckets();
            List<Bucket> bucketList = response.buckets();
            bucketList.forEach(bucket -> {
                System.out.println("Bucket Name: " + bucket.name());
            });

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListBuckets](#)”中的。

主题

- [操作](#)
- [场景](#)
- [无服务器示例](#)

操作

CopyObject

以下代码示例演示如何使用 CopyObject。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用 [S3Client](#) 复制对象。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CopyObject {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <objectKey> <fromBucket> <toBucket>

            Where:
                objectKey - The name of the object (for example, book.pdf).
                fromBucket - The S3 bucket name that contains the object (for
                example, bucket1).
                toBucket - The S3 bucket to copy the object to (for example,
                bucket2).

            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
    }

    String objectKey = args[0];
    String fromBucket = args[1];
    String toBucket = args[2];
    System.out.format("Copying object %s from bucket %s to %s\n", objectKey,
fromBucket, toBucket);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    copyBucketObject(s3, fromBucket, objectKey, toBucket);
    s3.close();
}

public static String copyBucketObject(S3Client s3, String fromBucket, String
objectKey, String toBucket) {
    CopyObjectRequest copyReq = CopyObjectRequest.builder()
        .sourceBucket(fromBucket)
        .sourceKey(objectKey)
        .destinationBucket(toBucket)
        .destinationKey(objectKey)
        .build();

    try {
        CopyObjectResponse copyRes = s3.copyObject(copyReq);
        return copyRes.copyObjectResult().toString();
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

使用 [S3 TransferManager](#) 将[对象从一个存储桶复制到另一个存储桶](#)。查看[完整文件](#)并[进行测试](#)。

```
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();
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CopyObject](#)”中的。

CreateBucket

以下代码示例演示如何使用 CreateBucket。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

创建存储桶。

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.HeadBucketResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class CreateBucket {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = ""

                Usage:
                <bucketName>\s

                Where:
                bucketName - The name of the bucket to create. The bucket name
                must be unique, or an error occurs.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Creating a bucket named %s\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();
    }
}
```



```
        createBucket(s3, bucketName);
        s3.close();
    }

    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                .bucket(bucketName)
                .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

创建一个启用对象锁定的存储桶。

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

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateBucket](#)”中的。

DeleteBucket

以下代码示例演示如何使用 DeleteBucket。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
    .bucket(bucket)
    .build();

s3.deleteBucket(deleteBucketRequest);
s3.close();
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteBucket](#)”中的。

DeleteBucketPolicy

以下代码示例演示如何使用 DeleteBucketPolicy。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteBucketPolicy {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <bucketName>

            Where:
                bucketName - The Amazon S3 bucket to delete the policy from (for
example, bucket1)."";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Deleting policy from bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
```

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

deleteS3BucketPolicy(s3, bucketName);
s3.close();
}

// Delete the bucket policy.
public static void deleteS3BucketPolicy(S3Client s3, String bucketName) {
    DeleteBucketPolicyRequest delReq = DeleteBucketPolicyRequest.builder()
        .bucket(bucketName)
        .build();

    try {
        s3.deleteBucketPolicy(delReq);
        System.out.println("Done!");
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteBucketPolicy](#)”中的。

DeleteBucketWebsite

以下代码示例演示如何使用 DeleteBucketWebsite。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

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

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

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = ""

                Usage:      <bucketName>

                Where:
                    bucketName - The Amazon S3 bucket to delete the website
configuration from.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Deleting website configuration for Amazon S3 bucket: %s
\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        deleteBucketWebsiteConfig(s3, bucketName);
        System.out.println("Done!");
        s3.close();
    }

    public static void deleteBucketWebsiteConfig(S3Client s3, String bucketName) {
        DeleteBucketWebsiteRequest delReq = DeleteBucketWebsiteRequest.builder()
                .bucket(bucketName)
```

```
        .build();

    try {
        s3.deleteBucketWebsite(delReq);
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.out.println("Failed to delete website configuration!");
        System.exit(1);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteBucketWebsite](#)”中的。

DeleteObjects

以下代码示例演示如何使用 DeleteObjects。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 */
```

```
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/

public class DeleteMultiObjects {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucketName>

            Where:
                bucketName - the Amazon S3 bucket name.
            "";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteBucketObjects(s3, bucketName);
        s3.close();
    }

    public static void deleteBucketObjects(S3Client s3, String bucketName) {
        // Upload three sample objects to the specified 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);
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteObjects](#)”中的。

GetBucketAcl

以下代码示例演示如何使用 GetBucketAcl。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAclRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAclResponse;
import software.amazon.awssdk.services.s3.model.Grant;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetAcl {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <objectKey>

                Where:
                bucketName - The Amazon S3 bucket to get the access control list
(ACL) for.
                objectKey - The object to get the ACL for.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String bucketName = args[0];
String objectKey = args[1];
System.out.println("Retrieving ACL for object: " + objectKey);
System.out.println("in bucket: " + bucketName);
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

getBucketACL(s3, objectKey, bucketName);
s3.close();
System.out.println("Done!");
}

public static String getBucketACL(S3Client s3, String objectKey, String
bucketName) {
    try {
        GetObjectAclRequest aclReq = GetObjectAclRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectAclResponse aclRes = s3.getObjectAcl(aclReq);
        List<Grant> grants = aclRes.grants();
        String grantee = "";
        for (Grant grant : grants) {
            System.out.format("  %s: %s\n", grant.grantee().id(),
grant.permission());
            grantee = grant.grantee().id();
        }

        return grantee;
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetBucketAcl](#)”中的。

GetBucketPolicy

以下代码示例演示如何使用 `GetBucketPolicy`。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetBucketPolicy {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName>

                Where:
                bucketName - The Amazon S3 bucket to get the policy from.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
String bucketName = args[0];
System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

String polText = getPolicy(s3, bucketName);
System.out.println("Policy Text: " + polText);
s3.close();
}

public static String getPolicy(S3Client s3, String bucketName) {
    String policyText;
    System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
    GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
        .bucket(bucketName)
        .build();

    try {
        GetBucketPolicyResponse policyRes = s3.getBucketPolicy(policyReq);
        policyText = policyRes.policy();
        return policyText;

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetBucketPolicy](#)”中的。

GetObject

以下代码示例演示如何使用 GetObject。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用 [S3Client](#) 以字节数组读取数据。

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectData {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName> <path>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
                path - The path where the file is written to.\s
            "";
```

```
        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String path = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getObjectBytes(s3, bucketName, keyName, path);
    }

    public static void getObjectBytes(S3Client s3, String bucketName, String
keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                .builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            ResponseBytes<GetObjectResponse> objectBytes =
s3.getObjectAsBytes(objectRequest);
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();

        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

使用 [S3 TransferManager](#) 将 S3 存储桶中的[对象下载](#)到本地文件。查看[完整文件](#)并[进行测试](#)。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;

    public Long downloadFile(S3TransferManager transferManager, String bucketName,
                            String key, String downloadedFileWithPath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
            .getObjectRequest(b -> b.bucket(bucketName).key(key))
            .destination(Paths.get(downloadedFileWithPath))
            .build();

        FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
        logger.info("Content length [{}]",
downloadResult.response().contentLength());
        return downloadResult.response().contentLength();
    }
```

使用 [S3Client](#) 读取属于对象的标签。

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tag;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectTags {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents the object.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listTags(s3, bucketName, keyName);
        s3.close();
    }
}
```



```
public static void listTags(S3Client s3, String bucketName, String keyName) {
    try {
        GetObjectTaggingRequest getTaggingRequest = GetObjectTaggingRequest
            .builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        GetObjectTaggingResponse tags = s3.getObjectTagging(getTaggingRequest);
        List<Tag> tagSet = tags.tagSet();
        for (Tag tag : tagSet) {
            System.out.println(tag.key());
            System.out.println(tag.value());
        }

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

使用 [S3 Client URL](#) 为对象获取 a。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetUrlRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.net.URL;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectUrl {
    public static void main(String[] args) {
        final String usage = ""
```

```
Usage:
    <bucketName> <keyName>\s

Where:
    bucketName - The Amazon S3 bucket name.
    keyName - A key name that represents the object.\s
""";

if (args.length != 2) {
    System.out.println(usage);
    System.exit(1);
}

String bucketName = args[0];
String keyName = args[1];
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

getURL(s3, bucketName, keyName);
s3.close();
}

public static void getURL(S3Client s3, String bucketName, String keyName) {
    try {
        GetUrlRequest request = GetUrlRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        URL url = s3.utilities().getUrl(request);
        System.out.println("The URL for " + keyName + " is " + url);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

使用 [S3Client](#) 通过 S3Presigner 客户端对象获取对象。

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.time.Duration;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.utils.IoUtils;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetObjectPresignedUrl {
    public static void main(String[] args) {
        final String USAGE = ""

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents a text file.\s

            """;

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Presigner presigner = S3Presigner.builder()
```

```
        .region(region)
        .build();

    getPresignedUrl(presigner, bucketName, keyName);
    presigner.close();
}

public static void getPresignedUrl(S3Presigner presigner, String bucketName,
String keyName) {
    try {
        GetObjectRequest getObjectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        GetObjectPresignRequest getObjectPresignRequest =
GetObjectPresignRequest.builder()
            .signatureDuration(Duration.ofMinutes(60))
            .getObjectRequest(getObjectRequest)
            .build();

        PresignedGetObjectRequest presignedGetObjectRequest =
presigner.presignGetObject(getObjectPresignRequest);
        String theUrl = presignedGetObjectRequest.url().toString();
        System.out.println("Presigned URL: " + theUrl);
        HttpURLConnection connection = (HttpURLConnection)
presignedGetObjectRequest.url().openConnection();
        presignedGetObjectRequest.httpRequest().headers().forEach((header,
values) -> {
            values.forEach(value -> {
                connection.addRequestProperty(header, value);
            });
        });

        // Send any request payload that the service needs (not needed when
// isBrowserExecutable is true).
        if (presignedGetObjectRequest.signedPayload().isPresent()) {
            connection.setDoOutput(true);

            try (InputStream signedPayload =
presignedGetObjectRequest.signedPayload().get().asInputStream();
                OutputStream httpOutputStream =
connection.getOutputStream()) {
                IOUtils.copy(signedPayload, httpOutputStream);
            }
        }
    }
}
```

```
        }
    }

    // Download the result of executing the request.
    try (InputStream content = connection.getInputStream()) {
        System.out.println("Service returned response: ");
        IoUtils.copy(content, System.out);
    }

} catch (S3Exception | IOException e) {
    e.printStackTrace();
}
}
}
```

使用对象和 [S3 Client](#) 获取 ResponseTransformer 对象。

```
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.sync.ResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetDataResponseTransformer {
    public static void main(String[] args) {
        final String usage = ""
```

```
Usage:
    <bucketName> <keyName> <path>

Where:
    bucketName - The Amazon S3 bucket name.\s
    keyName - The key name.\s
    path - The path where the file is written to.\s
""";

if (args.length != 3) {
    System.out.println(usage);
    System.exit(1);
}

String bucketName = args[0];
String keyName = args[1];
String path = args[2];
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

getObjectBytes(s3, bucketName, keyName, path);
s3.close();
}

public static void getObjectBytes(S3Client s3, String bucketName, String
keyName, String path) {
    try {
        GetObjectRequest objectRequest = GetObjectRequest
            .builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        ResponseBytes<GetObjectResponse> objectBytes =
s3.getObject(objectRequest, ResponseTransformer.toBytes());
        byte[] data = objectBytes.asByteArray();

        // Write the data to a local file.
        File myFile = new File(path);
        OutputStream os = new FileOutputStream(myFile);
        os.write(data);
        System.out.println("Successfully obtained bytes from an S3 object");
    }
}
```

```
        os.close();

    } catch (IOException ex) {
        ex.printStackTrace();
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetObject](#)”中的。

GetObjectLegalHold

以下代码示例演示如何使用 GetObjectLegalHold。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
// 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;
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetObjectLegalHold](#)”中的。

GetObjectLockConfiguration

以下代码示例演示如何使用 GetObjectLockConfiguration。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
// 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.getObjectLockConfiguration().getObjectLockEnabled());
    System.out.println("\tRule: "+
response.getObjectLockConfiguration().rule().defaultRetention());
}
```


- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetObjectLockConfiguration](#)”中的。

GetObjectRetention

以下代码示例演示如何使用 GetObjectRetention。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
// 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("Object 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;
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetObjectRetention](#)”中的。

HeadObject

以下代码示例演示如何使用 HeadObject。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

确定对象的内容类型。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetObjectContentType {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <keyName>>

                Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```

    }

    String bucketName = args[0];
    String keyName = args[1];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    getContentType(s3, bucketName, keyName);
    s3.close();
}

public static void getContentType(S3Client s3, String bucketName, String
keyName) {
    try {
        HeadObjectRequest objectRequest = HeadObjectRequest.builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        HeadObjectResponse objectHead = s3.headObject(objectRequest);
        String type = objectHead.contentType();
        System.out.println("The object content type is " + type);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
}

```

获取对象的还原状态。

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

public class GetObjectRestoreStatus {
    public static void main(String[] args) {

```

```
final String usage = ""

    Usage:
        <bucketName> <keyName>\s

    Where:
        bucketName - The Amazon S3 bucket name.\s
        keyName - A key name that represents the object.\s
    """;

if (args.length != 2) {
    System.out.println(usage);
    System.exit(1);
}

String bucketName = args[0];
String keyName = args[1];
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

checkStatus(s3, bucketName, keyName);
s3.close();
}

public static void checkStatus(S3Client s3, String bucketName, String keyName) {
    try {
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        HeadObjectResponse response = s3.headObject(headObjectRequest);
        System.out.println("The Amazon S3 object restoration status is " +
response.restore());

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [HeadObject](#)”中的。

ListBuckets

以下代码示例演示如何使用 ListBuckets。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListBuckets {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listAllBuckets(s3);
    }

    public static void listAllBuckets(S3Client s3) {
        ListBucketsResponse response = s3.listBuckets();
    }
}
```

```
List<Bucket> bucketList = response.buckets();
for (Bucket bucket: bucketList) {
    System.out.println("Bucket name "+bucket.name());
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListBuckets](#)”中的。

ListMultipartUploads

以下代码示例演示如何使用 ListMultipartUploads。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class ListMultipartUploads {
    public static void main(String[] args) {
        final String usage = ""
```

```
Usage:
    <bucketName>\s

Where:
    bucketName - The name of the Amazon S3 bucket where an in-
progress multipart upload is occurring.
    """;

if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

String bucketName = args[0];
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();
listUploads(s3, bucketName);
s3.close();
}

public static void listUploads(S3Client s3, String bucketName) {
    try {
        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
            .bucket(bucketName)
            .build();

        ListMultipartUploadsResponse response =
s3.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();
        for (MultipartUpload upload : uploads) {
            System.out.println("Upload in progress: Key = \"" + upload.key() +
"\", id = " + upload.uploadId());
        }

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListMultipartUploads](#)”中的。

ListObjectsV2

以下代码示例演示如何使用 ListObjectsV2。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class ListObjects {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName>\s

                Where:
                bucketName - The Amazon S3 bucket from which objects are read.\s
    }
```



```
        """);

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    listBucketObjects(s3, bucketName);
    s3.close();
}

public static void listBucketObjects(S3Client s3, String bucketName) {
    try {
        ListObjectsRequest listObjects = ListObjectsRequest
            .builder()
            .bucket(bucketName)
            .build();

        ListObjectsResponse res = s3.listObjects(listObjects);
        List<S3Object> objects = res.contents();
        for (S3Object myValue : objects) {
            System.out.print("\n The name of the key is " + myValue.key());
            System.out.print("\n The object is " + calKb(myValue.size()) + "
KBs");

            System.out.print("\n The owner is " + myValue.owner());
        }

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

// convert bytes to kbs.
private static long calKb(Long val) {
    return val / 1024;
}
}
```

使用分页列出对象。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;

public class ListObjectsPaginated {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName>\s

                Where:
                bucketName - The Amazon S3 bucket from which objects are read.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBucketObjects(s3, bucketName);
        s3.close();
    }

    public static void listBucketObjects(S3Client s3, String bucketName) {
        try {
            ListObjectsV2Request listReq = ListObjectsV2Request.builder()
                .bucket(bucketName)
                .maxKeys(1)
                .build();
        }
    }
}
```

```
ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);
listRes.stream()
    .flatMap(r -> r.contents().stream())
    .forEach(content -> System.out.println(" Key: " + content.key()
+ " size = " + content.size()));

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- 有关API详细信息，请参阅《AWS SDK for Java 2.x API参考资料》中的 [ListObjectsV2](#)。

PutBucketAcl

以下代码示例演示如何使用 PutBucketAcl。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AccessControlPolicy;
import software.amazon.awssdk.services.s3.model.Grant;
import software.amazon.awssdk.services.s3.model.Permission;
import software.amazon.awssdk.services.s3.model.PutBucketAclRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Type;

import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
```

```
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class SetAcl {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <id>\s

            Where:
                bucketName - The Amazon S3 bucket to grant permissions on.\s
                id - The ID of the owner of this bucket (you can get this value
from the AWS Management Console).
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String id = args[1];
        System.out.format("Setting access \n");
        System.out.println(" in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setBucketAcl(s3, bucketName, id);
        System.out.println("Done!");
        s3.close();
    }

    public static void setBucketAcl(S3Client s3, String bucketName, String id) {
        try {
            Grant ownerGrant = Grant.builder()
                .grantee(builder -> builder.id(id)
                    .type(Type.CANONICAL_USER))
                .permission(Permission.FULL_CONTROL)

```

```
        .build();

        List<Grant> grantList2 = new ArrayList<>();
        grantList2.add(ownerGrant);

        AccessControlPolicy acl = AccessControlPolicy.builder()
            .owner(builder -> builder.id(id))
            .grants(grantList2)
            .build();

        PutBucketAclRequest putAclReq = PutBucketAclRequest.builder()
            .bucket(bucketName)
            .accessControlPolicy(acl)
            .build();

        s3.putBucketAcl(putAclReq);

    } catch (S3Exception e) {
        e.printStackTrace();
        System.exit(1);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [PutBucketAcl](#)”中的。

PutBucketCors

以下代码示例演示如何使用 PutBucketCors。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import java.util.ArrayList;
```

```
import java.util.List;
import software.amazon.awssdk.services.s3.model.GetBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.GetBucketCorsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.CORSRule;
import software.amazon.awssdk.services.s3.model.CORSConfiguration;
import software.amazon.awssdk.services.s3.model.PutBucketCorsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class S3Cors {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <accountId>\s

                Where:
                bucketName - The Amazon S3 bucket to upload an object into.
                accountId - The id of the account that owns the Amazon S3
bucket.

                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String accountId = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setCorsInformation(s3, bucketName, accountId);
        getBucketCorsInformation(s3, bucketName, accountId);
    }
}
```

```
        deleteBucketCorsInformation(s3, bucketName, accountId);
        s3.close();
    }

    public static void deleteBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
        try {
            DeleteBucketCorsRequest bucketCorsRequest =
DeleteBucketCorsRequest.builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            s3.deleteBucketCors(bucketCorsRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
        try {
            GetBucketCorsRequest bucketCorsRequest = GetBucketCorsRequest.builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            GetBucketCorsResponse corsResponse =
s3.getBucketCors(bucketCorsRequest);
            List<CORSRule> corsRules = corsResponse.getCORSRules();
            for (CORSRule rule : corsRules) {
                System.out.println("allowOrigins: " + rule.allowedOrigins());
                System.out.println("AllowedMethod: " + rule.allowedMethods());
            }

        } catch (S3Exception e) {

            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

```
public static void setCorsInformation(S3Client s3, String bucketName, String
accountId) {
    List<String> allowMethods = new ArrayList<>();
    allowMethods.add("PUT");
    allowMethods.add("POST");
    allowMethods.add("DELETE");

    List<String> allowOrigins = new ArrayList<>();
    allowOrigins.add("http://example.com");
    try {
        // Define CORS rules.
        CORSRule corsRule = CORSRule.builder()
            .allowedMethods(allowMethods)
            .allowedOrigins(allowOrigins)
            .build();

        List<CORSRule> corsRules = new ArrayList<>();
        corsRules.add(corsRule);
        CORSConfiguration configuration = CORSConfiguration.builder()
            .corsRules(corsRules)
            .build();

        PutBucketCorsRequest putBucketCorsRequest =
PutBucketCorsRequest.builder()
            .bucket(bucketName)
            .corsConfiguration(configuration)
            .expectedBucketOwner(accountId)
            .build();

        s3.putBucketCors(putBucketCorsRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [PutBucketCors](#)”中的。

PutBucketLifecycleConfiguration

以下代码示例演示如何使用 PutBucketLifecycleConfiguration。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.Transition;
import
    software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationRequest;
import
    software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketLifecycleRequest;
import software.amazon.awssdk.services.s3.model.TransitionStorageClass;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import
    software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class LifecycleConfiguration {
    public static void main(String[] args) {
        final String usage = ""
```

```

Usage:
    <bucketName> <accountId>\s

Where:
    bucketName - The Amazon Simple Storage Service
(Amazon S3) bucket to upload an object into.
    accountId - The id of the account that owns the
Amazon S3 bucket.

    """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String accountId = args[1];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    setLifecycleConfig(s3, bucketName, accountId);
    getLifecycleConfig(s3, bucketName, accountId);
    deleteLifecycleConfig(s3, bucketName, accountId);
    System.out.println("You have successfully created, updated, and
deleted a Lifecycle configuration");
    s3.close();
}

    public static void setLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
        try {
            // Create a rule to archive objects with the
"glacierobjects/" prefix to Amazon
            // S3 Glacier.
            LifecycleRuleFilter ruleFilter =
LifecycleRuleFilter.builder()
                .prefix("glacierobjects/")
                .build();

            Transition transition = Transition.builder()

```

```
.storageClass(TransitionStorageClass.GLACIER)
    .days(0)
    .build();

LifecycleRule rule1 = LifecycleRule.builder()
    .id("Archive immediately rule")
    .filter(ruleFilter)
    .transitions(transition)
    .status(ExpirationStatus.ENABLED)
    .build();

// Create a second rule.
Transition transition2 = Transition.builder()

.storageClass(TransitionStorageClass.GLACIER)
    .days(0)
    .build();

List<Transition> transitionList = new ArrayList<>();
transitionList.add(transition2);

LifecycleRuleFilter ruleFilter2 =
LifecycleRuleFilter.builder()
    .prefix("glacierobjects/")
    .build();

LifecycleRule rule2 = LifecycleRule.builder()
    .id("Archive and then delete rule")
    .filter(ruleFilter2)
    .transitions(transitionList)
    .status(ExpirationStatus.ENABLED)
    .build();

// Add the LifecycleRule objects to an ArrayList.
ArrayList<LifecycleRule> ruleList = new ArrayList<>();
ruleList.add(rule1);
ruleList.add(rule2);

BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
    .rules(ruleList)
    .build();
```

```
        PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
        .builder()
        .bucket(bucketName)

        .lifecycleConfiguration(lifecycleConfiguration)
        .expectedBucketOwner(accountId)
        .build();

s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Retrieve the configuration and add a new rule.
    public static void getLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
        try {
            GetBucketLifecycleConfigurationRequest
getBucketLifecycleConfigurationRequest = GetBucketLifecycleConfigurationRequest
            .builder()
            .bucket(bucketName)
            .expectedBucketOwner(accountId)
            .build();

            GetBucketLifecycleConfigurationResponse response = s3

.getBucketLifecycleConfiguration(getBucketLifecycleConfigurationRequest);
            List<LifecycleRule> newList = new ArrayList<>();
            List<LifecycleRule> rules = response.rules();
            for (LifecycleRule rule : rules) {
                newList.add(rule);
            }

            // Add a new rule with both a prefix predicate and a tag
predicate.

            LifecycleRuleFilter ruleFilter =
LifecycleRuleFilter.builder()
                .prefix("YearlyDocuments/")
                .build();
```

```
        Transition transition = Transition.builder()

        .storageClass(TransitionStorageClass.GLACIER)
            .days(3650)
            .build();

        LifecycleRule rule1 = LifecycleRule.builder()
            .id("NewRule")
            .filter(ruleFilter)
            .transitions(transition)
            .status(ExpirationStatus.ENABLED)
            .build();

        // Add the new rule to the list.
        newList.add(rule1);
        BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
            .rules(newList)
            .build();

        PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
            .builder()
            .bucket(bucketName)

        .lifecycleConfiguration(lifecycleConfiguration)
            .expectedBucketOwner(accountId)
            .build();

s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Delete the configuration from the Amazon S3 bucket.
    public static void deleteLifecycleConfig(S3Client s3, String bucketName,
String accountId) {
        try {
```

```
        DeleteBucketLifecycleRequest deleteBucketLifecycleRequest =
DeleteBucketLifecycleRequest
                                .builder()
                                .bucket(bucketName)
                                .expectedBucketOwner(accountId)
                                .build();

        s3.deleteBucketLifecycle(deleteBucketLifecycleRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [PutBucketLifecycleConfiguration](#)”中的。

PutBucketPolicy

以下代码示例演示如何使用 PutBucketPolicy。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.List;
```

```
import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetBucketPolicy {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <polFile>

            Where:
                bucketName - The Amazon S3 bucket to set the policy on.
                polFile - A JSON file containing the policy (see the Amazon S3
Readme for an example).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String polFile = args[1];
        String policyText = getBucketPolicyFromFile(polFile);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setPolicy(s3, bucketName, policyText);
        s3.close();
    }

    public static void setPolicy(S3Client s3, String bucketName, String policyText)
    {
        System.out.println("Setting policy:");
    }
}
```

```
System.out.println("----");
System.out.println(policyText);
System.out.println("----");
System.out.format("On Amazon S3 bucket: \"%s\"\n", bucketName);

try {
    PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
        .bucket(bucketName)
        .policy(policyText)
        .build();

    s3.putBucketPolicy(policyReq);

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

System.out.println("Done!");
}

// Loads a JSON-formatted policy from a file
public static String getBucketPolicyFromFile(String policyFile) {

    StringBuilder fileText = new StringBuilder();
    try {
        List<String> lines = Files.readAllLines(Paths.get(policyFile),
StandardCharsets.UTF_8);
        for (String line : lines) {
            fileText.append(line);
        }

    } catch (IOException e) {
        System.out.format("Problem reading file: \"%s\"", policyFile);
        System.out.println(e.getMessage());
    }

    try {
        final JsonParser parser = new
ObjectMapper().getFactory().createParser(fileText.toString());
        while (parser.nextToken() != null) {
        }

    } catch (IOException jpe) {
```



```
        jpe.printStackTrace();
    }
    return fileText.toString();
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [PutBucketPolicy](#)”中的。

PutBucketWebsite

以下代码示例演示如何使用 PutBucketWebsite。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.IndexDocument;
import software.amazon.awssdk.services.s3.model.PutBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.WebsiteConfiguration;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class SetWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = ""

                Usage:    <bucketName> [indexdoc]\s
```

```

        Where:
            bucketName    - The Amazon S3 bucket to set the website
configuration on.\s
            indexdoc     - The index document, ex. 'index.html'
                        If not specified, 'index.html' will be set.
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String indexDoc = "index.html";
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    setWebsiteConfig(s3, bucketName, indexDoc);
    s3.close();
}

public static void setWebsiteConfig(S3Client s3, String bucketName, String
indexDoc) {
    try {
        WebsiteConfiguration websiteConfig = WebsiteConfiguration.builder()
            .indexDocument(IndexDocument.builder().suffix(indexDoc).build())
            .build();

        PutBucketWebsiteRequest pubWebsiteReq =
PutBucketWebsiteRequest.builder()
            .bucket(bucketName)
            .websiteConfiguration(websiteConfig)
            .build();

        s3.putBucketWebsite(pubWebsiteReq);
        System.out.println("The call was successful");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

```
}  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [PutBucketWebsite](#)”中的。

PutObject

以下代码示例演示如何使用 PutObject。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用 [S3Client](#) 将文件上传到存储桶。

```
import software.amazon.awssdk.core.sync.RequestBody;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.PutObjectRequest;  
import software.amazon.awssdk.services.s3.model.S3Exception;  
import java.io.File;  
import java.util.HashMap;  
import java.util.Map;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 *  
 * For more information, see the following documentation topic:  
 *  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
  
public class PutObject {  
    public static void main(String[] args) {  
        final String usage = ""
```

Usage:

```
<bucketName> <objectKey> <objectPath>\s
```

Where:

bucketName - The Amazon S3 bucket to upload an object into.

objectKey - The object to upload (for example, book.pdf).

objectPath - The path where the file is located (for example, C:/

AWS/book2.pdf).\s

```
""";
```

```
if (args.length != 3) {  
    System.out.println(usage);  
    System.exit(1);  
}
```

```
String bucketName = args[0];  
String objectKey = args[1];  
String objectPath = args[2];  
Region region = Region.US_EAST_1;  
S3Client s3 = S3Client.builder()  
    .region(region)  
    .build();
```

```
putS3Object(s3, bucketName, objectKey, objectPath);  
s3.close();
```

```
}
```

```
// This example uses RequestBody.fromFile to avoid loading the whole file into  
// memory.
```

```
public static void putS3Object(S3Client s3, String bucketName, String objectKey,  
String objectPath) {
```

```
    try {  
        Map<String, String> metadata = new HashMap<>();  
        metadata.put("x-amz-meta-myVal", "test");  
        PutObjectRequest putOb = PutObjectRequest.builder()  
            .bucket(bucketName)  
            .key(objectKey)  
            .metadata(metadata)  
            .build();
```

```
        s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));  
        System.out.println("Successfully placed " + objectKey + " into bucket "  
+ bucketName);
```

```

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}

```

使用 [S 3](#) 将[文件上传TransferManager](#)到存储桶。查看[完整文件](#)并[进行测试](#)。

```

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();
    }

```

使用 [S3Client](#) 将对象上传到存储桶并设置标签。

```

    public static void putS3ObjectTags(S3Client s3, String bucketName, String
    objectKey, String objectPath) {
        try {
            Tag tag1 = Tag.builder()

```

```
        .key("Tag 1")
        .value("This is tag 1")
        .build();

    Tag tag2 = Tag.builder()
        .key("Tag 2")
        .value("This is tag 2")
        .build();

    List<Tag> tags = new ArrayList<>();
    tags.add(tag1);
    tags.add(tag2);

    Tagging allTags = Tagging.builder()
        .tagSet(tags)
        .build();

    PutObjectRequest putOb = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .tagging(allTags)
        .build();

    s3.putObject(putOb, RequestBody.fromBytes(getObjectFile(objectPath)));

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void updateObjectTags(S3Client s3, String bucketName, String
objectKey) {
    try {
        GetObjectTaggingRequest taggingRequest =
GetObjectTaggingRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectTaggingResponse getTaggingRes =
s3.getObjectTagging(taggingRequest);
        List<Tag> obTags = getTaggingRes.tagSet();
        for (Tag sinTag : obTags) {
```

```
        System.out.println("The tag key is: " + sinTag.key());
        System.out.println("The tag value is: " + sinTag.value());
    }

    // Replace the object's tags with two new tags.
    Tag tag3 = Tag.builder()
        .key("Tag 3")
        .value("This is tag 3")
        .build();

    Tag tag4 = Tag.builder()
        .key("Tag 4")
        .value("This is tag 4")
        .build();

    List<Tag> tags = new ArrayList<>();
    tags.add(tag3);
    tags.add(tag4);

    Tagging updatedTags = Tagging.builder()
        .tagSet(tags)
        .build();

    PutObjectTaggingRequest taggingRequest1 =
    PutObjectTaggingRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .tagging(updatedTags)
        .build();

    s3.putObjectTagging(taggingRequest1);
    GetObjectTaggingResponse getTaggingRes2 =
    s3.getObjectTagging(taggingRequest);
    List<Tag> modTags = getTaggingRes2.tagSet();
    for (Tag sinTag : modTags) {
        System.out.println("The tag key is: " + sinTag.key());
        System.out.println("The tag value is: " + sinTag.value());
    }

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

```
// Return a byte array.
private static byte[] getObjectFile(String filePath) {
    FileInputStream fileInputStream = null;
    byte[] byteArray = null;

    try {
        File file = new File(filePath);
        byteArray = new byte[(int) file.length()];
        fileInputStream = new FileInputStream(file);
        fileInputStream.read(byteArray);

    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (fileInputStream != null) {
            try {
                fileInputStream.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    return byteArray;
}
}
```

使用 [S3Client](#) 将对象上传到存储桶并设置元数据。

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.File;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```



```
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class PutObjectMetadata {
    public static void main(String[] args) {
        final String USAGE = ""

            Usage:
                <bucketName> <objectKey> <objectPath>\s

            Where:
                bucketName - The Amazon S3 bucket to upload an object into.
                objectKey - The object to upload (for example, book.pdf).
                objectPath - The path where the file is located (for example, C:/
AWS/book2.pdf).\s
                """;

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        String objectPath = args[2];
        System.out.println("Putting object " + objectKey + " into bucket " +
bucketName);
        System.out.println(" in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        putS3Object(s3, bucketName, objectKey, objectPath);
        s3.close();
    }

    // This example uses RequestBody.fromFile to avoid loading the whole file into
    // memory.
    public static void putS3Object(S3Client s3, String bucketName, String objectKey,
String objectPath) {
        try {
```

```

    Map<String, String> metadata = new HashMap<>();
    metadata.put("author", "Mary Doe");
    metadata.put("version", "1.0.0.0");

    PutObjectRequest putOb = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .metadata(metadata)
        .build();

    s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
    System.out.println("Successfully placed " + objectKey + " into bucket "
+ bucketName);

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
}

```

使用 [S3Client](#) 将对象上传到存储桶并设置对象保留值。

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

```

```
public class PutObjectRetention {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <key> <bucketName>\s

            Where:
                key - The name of the object (for example, book.pdf).\s
                bucketName - The Amazon S3 bucket name that contains the object
(for example, bucket1).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String key = args[0];
        String bucketName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setRetentionPeriod(s3, key, bucketName);
        s3.close();
    }

    public static void setRetentionPeriod(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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [PutObject](#)”中的。

PutObjectLegalHold

以下代码示例演示如何使用 PutObjectLegalHold。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
// 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
+".");
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [PutObjectLegalHold](#)”中的。

PutObjectLockConfiguration

以下代码示例演示如何使用 PutObjectLockConfiguration。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

设置存储桶的对象锁定配置。

```
// 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() +
    "'");
}
}

```

设置存储桶的默认保留期。

```

// 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 retention = DefaultRetention.builder()
    .days(1)
    .mode(ObjectLockRetentionMode.GOVERNANCE)
    .build();

ObjectLockRule lockRule = ObjectLockRule.builder()
    .defaultRetention(retention)
    .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 +".");
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [PutObjectLockConfiguration](#)”中的。

PutObjectRetention

以下代码示例演示如何使用 PutObjectRetention。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
// 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 +".");
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [PutObjectRetention](#)”中的。

RestoreObject

以下代码示例演示如何使用 RestoreObject。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.RestoreRequest;
import software.amazon.awssdk.services.s3.model.GlacierJobParameters;
import software.amazon.awssdk.services.s3.model.RestoreObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tier;

/*
 * For more information about restoring an object, see "Restoring an archived
 * object" at
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/restoring-objects.html
 *
 * Before running this Java V2 code example, set up your development environment,
 * including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RestoreObject {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <bucketName> <keyName> <expectedBucketOwner>

                Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name of an object with a Storage class value
of Glacier.\s
                expectedBucketOwner - The account that owns the bucket (you can
obtain this value from the AWS Management Console).\s
                """;
```

```
    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String keyName = args[1];
    String expectedBucketOwner = args[2];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    restoreS3Object(s3, bucketName, keyName, expectedBucketOwner);
    s3.close();
}

public static void restoreS3Object(S3Client s3, String bucketName, String
keyName, String expectedBucketOwner) {
    try {
        RestoreRequest restoreRequest = RestoreRequest.builder()
            .days(10)

.glacierJobParameters(GlacierJobParameters.builder().tier(Tier.STANDARD).build())
            .build();

        RestoreObjectRequest objectRequest = RestoreObjectRequest.builder()
            .expectedBucketOwner(expectedBucketOwner)
            .bucket(bucketName)
            .key(keyName)
            .restoreRequest(restoreRequest)
            .build();

        s3.restoreObject(objectRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [RestoreObject](#)”中的。

SelectObjectContent

以下代码示例演示如何使用 SelectObjectContent。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

以下示例显示了使用JSON对象的查询。[完整的示例](#)还显示了CSV对象的用法。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.CSVInput;
import software.amazon.awssdk.services.s3.model.CSVOutput;
import software.amazon.awssdk.services.s3.model.CompressionType;
import software.amazon.awssdk.services.s3.model.ExpressionType;
import software.amazon.awssdk.services.s3.model.FileHeaderInfo;
import software.amazon.awssdk.services.s3.model.InputSerialization;
import software.amazon.awssdk.services.s3.model.JSONInput;
import software.amazon.awssdk.services.s3.model.JSONOutput;
import software.amazon.awssdk.services.s3.model.JSONType;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.OutputSerialization;
import software.amazon.awssdk.services.s3.model.Progress;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.SelectObjectContentRequest;
import software.amazon.awssdk.services.s3.model.SelectObjectContentResponseHandler;
import software.amazon.awssdk.services.s3.model.Stats;

import java.io.IOException;
import java.net.URL;
```

```
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

public class SelectObjectContentExample {
    static final Logger logger =
    LoggerFactory.getLogger(SelectObjectContentExample.class);
    static final String BUCKET_NAME = "select-object-content-" + UUID.randomUUID();
    static final S3AsyncClient s3AsyncClient = S3AsyncClient.create();
    static String FILE_CSV = "csv";
    static String FILE_JSON = "json";
    static String URL_CSV = "https://raw.githubusercontent.com/mledoze/countries/
master/dist/countries.csv";
    static String URL_JSON = "https://raw.githubusercontent.com/mledoze/countries/
master/dist/countries.json";

    public static void main(String[] args) {
        SelectObjectContentExample selectObjectContentExample = new
SelectObjectContentExample();
        try {
            SelectObjectContentExample.setUp();
            selectObjectContentExample.runSelectObjectContentMethodForJSON();
            selectObjectContentExample.runSelectObjectContentMethodForCSV();
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
            System.exit(1);
        } finally {
            SelectObjectContentExample.tearDown();
        }
    }

    EventStreamInfo runSelectObjectContentMethodForJSON() {
        // Set up request parameters.
        final String queryExpression = "select * from s3object[*][*] c where c.area
< 350000";
        final String fileType = FILE_JSON;

        InputSerialization inputSerialization = InputSerialization.builder()
            .json(JSONInput.builder().type(JSONType.DOCUMENT).build())
            .compressionType(CompressionType.NONE)
            .build();

        OutputSerialization outputSerialization = OutputSerialization.builder()
```

```

        .json(JSONOutput.builder().recordDelimiter(null).build())
        .build();

// Build the SelectObjectContentRequest.
SelectObjectContentRequest select = SelectObjectContentRequest.builder()
    .bucket(BUCKET_NAME)
    .key(FILE_JSON)
    .expression(queryExpression)
    .expressionType(ExpressionType.SQL)
    .inputSerialization(inputSerialization)
    .outputSerialization(outputSerialization)
    .build();

EventStreamInfo eventStreamInfo = new EventStreamInfo();
// Call the selectObjectContent method with the request and a response
handler.
// Supply an EventStreamInfo object to the response handler to gather
records and information from the response.
s3AsyncClient.selectObjectContent(select,
buildResponseHandler(eventStreamInfo)).join();

// Log out information gathered while processing the response stream.
long recordCount = eventStreamInfo.getRecords().stream().mapToInt(record ->
    record.split("\n").length
).sum();
logger.info("Total records {}: {}", fileType, recordCount);
logger.info("Visitor onRecords for fileType {} called {} times", fileType,
eventStreamInfo.getCountOnRecordsCalled());
logger.info("Visitor onStats for fileType {}, {}", fileType,
eventStreamInfo.getStats());
logger.info("Visitor onContinuations for fileType {}, {}", fileType,
eventStreamInfo.getCountContinuationEvents());
return eventStreamInfo;
}

static SelectObjectContentResponseHandler buildResponseHandler(EventStreamInfo
eventStreamInfo) {
// Use a Visitor to process the response stream. This visitor logs
information and gathers details while processing.
final SelectObjectContentResponseHandler.Visitor visitor =
SelectObjectContentResponseHandler.Visitor.builder()
    .onRecords(r -> {
        logger.info("Record event received.");
        eventStreamInfo.addRecord(r.payload().asUtf8String());
    });
}

```

```

        eventStreamInfo.incrementOnRecordsCalled();
    })
    .onCont(ce -> {
        logger.info("Continuation event received.");
        eventStreamInfo.incrementContinuationEvents();
    })
    .onProgress(pe -> {
        Progress progress = pe.details();
        logger.info("Progress event received:\n bytesScanned:
{} \n bytesProcessed: {} \n bytesReturned: {}",
            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;
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [SelectObjectContent](#)”中的。

场景

创建预签名 URL

以下代码示例演示如何为 Amazon S3 创建URL预签名文件并上传对象。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

URL为对象生成预签名文件，然后将其下载（GET请求）。

导入。

```
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;
```

生成URL。

```
/* Create a pre-signed URL to download an object in a subsequent GET request. */
public String createPresignedGetUrl(String bucketName, String keyName) {
    try (S3Presigner presigner = S3Presigner.create()) {

        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();
```



```

        GetObjectPresignRequest presignRequest =
GetObjectPresignRequest.builder()
            .signatureDuration(Duration.ofMinutes(10)) // The URL will
expire in 10 minutes.
            .getObjectRequest(objectRequest)
            .build();

        PresignedGetObjectRequest presignedRequest =
presigner.presignGetObject(presignRequest);
        logger.info("Presigned URL: [{}]", presignedRequest.url().toString());
        logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}

```

使用以下三种方法中的任何一种下载对象。

使用 `JDKURLConnection` (自 v1.1 起) 类进行下载。

```

/* 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();
}

```

使用 `JDKHttpClient` (自 v11 起) 类进行下载。

```
/* 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();
}
```

使用 AWS SDK 适用于 Java 的 `SdkHttpClient` 类进行下载。

```
/* Use the AWS SDK for Java SdkHttpClient class to do the download. */
public byte[] useSdkHttpClientToPut(String presignedUrlString) {

    ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.
    try {
        URL presignedUrl = new URL(presignedUrlString);
        SdkHttpRequest request = SdkHttpRequest.builder()
            .method(SdkHttpMethod.GET)
            .uri(presignedUrl.toURI())
            .build();
```

```

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
            response.responseBody().ifPresentOrElse(
                abortableInputStream -> {
                    try {
                        IoUtils.copy(abortableInputStream,
byteArrayOutputStream);
                    } catch (IOException e) {
                        throw new RuntimeException(e);
                    }
                },
                () -> logger.error("No response body."));

            logger.info("HTTP Response code is {}",
response.httpResponse().statusCode());
        }
    } catch (URISyntaxException | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}

```

URL为上传生成预签名文件，然后上传文件（PUT请求）。

导入。

```

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;
```

生成URL.

```
/* Create a presigned URL to use in a subsequent PUT request */
public String createPresignedUrl(String bucketName, String keyName, Map<String,
String> metadata) {
    try (S3Presigner presigner = S3Presigner.create()) {

        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .metadata(metadata)
            .build();

        PutObjectPresignRequest presignRequest =
PutObjectPresignRequest.builder()
            .signatureDuration(Duration.ofMinutes(10)) // The URL expires
in 10 minutes.
            .putObjectRequest(objectRequest)
            .build();
```

```

        PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignedRequest);
        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();
    }
}

```

使用以下三种方法中的任何一种上传文件对象。

使用 `JDKURLConnection` (自 v1.1 起) 类进行上传。

```

/* 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);
}
}

```

使用 `JDKHttpClient` (自 v11 起) 类进行上传。

```

/* 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);
    }
}

```

使用 AWS 适用于 Java 的 `V2 SdkHttpClient` 类进行上传。

```

/* 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);
    }
}
```

删除未完成的分段上传

以下代码示例显示如何删除或停止未完成的 Amazon S3 分段上传。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

要停止正在进行或由于任何原因而未完成的分段上传，您可以获取上传列表，然后删除这些上传，如以下示例所示。

```
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);
        }
    }
}
```

要删除在某个日期之前或之后启动的未完成分段上传，您可以根据某个时间点有选择地删除分段上传，如以下示例所示。


```

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);
            }
        }
    }
}
}
}

```

如果您在开始分段上传后可以访问上传 ID，则可以使用该 ID 删除正在进行的上传。

```

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);
    }
}

```

要一致地删除超过一定天数的未完成分段上传，请为存储桶设置存储桶生命周期配置。以下示例显示如何创建一条规则来删除超过 7 天的未完成上传。

```

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());
    }
}
}

```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [AbortMultipartUpload](#)
 - [ListMultipartUploads](#)
 - [PutBucketLifecycleConfiguration](#)

将对象下载到本地目录

以下代码示例演示了如何将 Amazon Simple Storage Service (Amazon S3) 桶中的所有对象下载到本地目录。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用 [S3 TransferManager](#) 将[所有 S3 对象下载](#)到同一 S3 存储桶中。查看[完整文件](#)并[进行测试](#)。

```
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();
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DownloadDirectory](#)”中的。

桶和对象入门

以下代码示例展示了如何：

- 创建桶并将文件上载到其中。
- 从桶中下载对象。
- 将对象复制到存储桶中的子文件夹。
- 列出存储桶中的对象。
- 删除存储桶及其对象。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code example performs the following tasks:
 *
 * 1. Creates an Amazon S3 bucket.
```

- * 2. Uploads an object to the bucket.
 - * 3. Downloads the object to another local file.
 - * 4. Uploads an object using multipart upload.
 - * 5. List all objects located in the Amazon S3 bucket.
 - * 6. Copies the object to another Amazon S3 bucket.
 - * 7. Deletes the object from the Amazon S3 bucket.
 - * 8. Deletes the Amazon S3 bucket.
- */

```
public class S3Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException {
        final String usage = ""

            Usage:
                <bucketName> <key> <objectPath> <savePath> <toBucket>

            Where:
                bucketName - The Amazon S3 bucket to create.
                key - The key to use.
                objectPath - The path where the file is located (for example,
C:/AWS/book2.pdf).
                savePath - The path where the file is saved after it's
downloaded (for example, C:/AWS/book2.pdf).
                toBucket - An Amazon S3 bucket to where an object is copied to
(for example, C:/AWS/book2.pdf).\s
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String key = args[1];
        String objectPath = args[2];
        String savePath = args[3];
        String toBucket = args[4];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();
```

```
System.out.println(DASHES);
System.out.println("Welcome to the Amazon S3 example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create an Amazon S3 bucket.");
createBucket(s3, bucketName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Update a local file to the Amazon S3 bucket.");
uploadLocalFile(s3, bucketName, key, objectPath);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Download the object to another local file.");
getObjectBytes(s3, bucketName, key, savePath);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Perform a multipart upload.");
String multipartKey = "multiPartKey";
multipartUpload(s3, toBucket, multipartKey);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. List all objects located in the Amazon S3 bucket.");
listAllObjects(s3, bucketName);
anotherListExample(s3, bucketName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Copy the object to another Amazon S3 bucket.");
copyBucketObject(s3, bucketName, key, toBucket);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Delete the object from the Amazon S3 bucket.");
deleteObjectFromBucket(s3, bucketName, key);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Delete the Amazon S3 bucket.");
deleteBucket(s3, bucketName);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("All Amazon S3 operations were successfully performed");
        System.out.println(DASHES);
        s3.close();
    }

    // Create a bucket by using a S3Waiter object.
    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                .bucket(bucketName)
                .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteBucket(S3Client client, String bucket) {
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
            .bucket(bucket)
            .build();

        client.deleteBucket(deleteBucketRequest);
        System.out.println(bucket + " was deleted.");
    }

    /**
     * Upload an object in parts.
```

```
    */
    public static void multipartUpload(S3Client s3, String bucketName, String key) {
        int mB = 1024 * 1024;
        // First create a multipart upload and get the upload id.
        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        CreateMultipartUploadResponse response =
s3.createMultipartUpload(createMultipartUploadRequest);
        String uploadId = response.uploadId();
        System.out.println(uploadId);

        // Upload all the different parts of the object.
        UploadPartRequest uploadPartRequest1 = UploadPartRequest.builder()
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)
            .partNumber(1).build();

        String etag1 = s3.uploadPart(uploadPartRequest1,
RequestBody.fromByteBuffer(getRandomByteBuffer(5 * mB)))
            .eTag();
        CompletedPart part1 =
CompletedPart.builder().partNumber(1).eTag(etag1).build();

        UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder().bucket(bucketName).key(key)
            .uploadId(uploadId)
            .partNumber(2).build();
        String etag2 = s3.uploadPart(uploadPartRequest2,
RequestBody.fromByteBuffer(getRandomByteBuffer(3 * mB)))
            .eTag();
        CompletedPart part2 =
CompletedPart.builder().partNumber(2).eTag(etag2).build();

        // Call completeMultipartUpload operation to tell S3 to merge all uploaded
        // parts and finish the multipart operation.
        CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
            .parts(part1, part2)
            .build();
    }
}
```



```
        CompleteMultipartUploadRequest completeMultipartUploadRequest =
CompleteMultipartUploadRequest.builder()
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)
    .multipartUpload(completedMultipartUpload)
    .build();

    s3.completeMultipartUpload(completeMultipartUploadRequest);
}

private static ByteBuffer getRandomByteBuffer(int size) {
    byte[] b = new byte[size];
    new Random().nextBytes(b);
    return ByteBuffer.wrap(b);
}

public static void getObjectBytes(S3Client s3, String bucketName, String
keyName, String path) {
    try {
        GetObjectRequest objectRequest = GetObjectRequest
            .builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        ResponseBytes<GetObjectResponse> objectBytes =
s3.getObjectAsBytes(objectRequest);
        byte[] data = objectBytes.asByteArray();

        // Write the data to a local file.
        File myFile = new File(path);
        OutputStream os = new FileOutputStream(myFile);
        os.write(data);
        System.out.println("Successfully obtained bytes from an S3 object");
        os.close();

    } catch (IOException ex) {
        ex.printStackTrace();
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}

    public static void uploadLocalFile(S3Client s3, String bucketName, String key,
String objectPath) {
        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        s3.putObject(objectRequest, RequestBody.fromFile(new File(objectPath)));
    }

    public static void listAllObjects(S3Client s3, String bucketName) {
        ListObjectsV2Request listObjectsReqManual = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .maxKeys(1)
            .build();

        boolean done = false;
        while (!done) {
            ListObjectsV2Response listObjResponse =
s3.listObjectsV2(listObjectsReqManual);
            for (S3Object content : listObjResponse.contents()) {
                System.out.println(content.key());
            }

            if (listObjResponse.nextContinuationToken() == null) {
                done = true;
            }

            listObjectsReqManual = listObjectsReqManual.toBuilder()
                .continuationToken(listObjResponse.nextContinuationToken())
                .build();
        }
    }

    public static void anotherListExample(S3Client s3, String bucketName) {
        ListObjectsV2Request listReq = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .maxKeys(1)
            .build();

        ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);
    }
}
```

```
// Process response pages.
listRes.stream()
    .flatMap(r -> r.contents().stream())
    .forEach(content -> System.out.println(" Key: " + content.key() + "
size = " + content.size()));

// Helper method to work with paginated collection of items directly.
listRes.contents().stream()
    .forEach(content -> System.out.println(" Key: " + content.key() + "
size = " + content.size()));

for (S3Object content : listRes.contents()) {
    System.out.println(" Key: " + content.key() + " size = " +
content.size());
}
}

public static void deleteObjectFromBucket(S3Client s3, String bucketName, String
key) {
    DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    s3.deleteObject(deleteObjectRequest);
    System.out.println(key + " was deleted");
}

public static String copyBucketObject(S3Client s3, String fromBucket, String
objectKey, String toBucket) {
    String encodedUrl = null;
    try {
        encodedUrl = URLEncoder.encode(fromBucket + "/" + objectKey,
StandardCharsets.UTF_8.toString());
    } catch (UnsupportedEncodingException e) {
        System.out.println("URL could not be encoded: " + e.getMessage());
    }
    CopyObjectRequest copyReq = CopyObjectRequest.builder()
        .copySource(encodedUrl)
        .destinationBucket(toBucket)
        .destinationKey(objectKey)
        .build();

    try {
```

```
CopyObjectResponse copyRes = s3.copyObject(copyReq);
System.out.println("The " + objectKey + " was copied to " + toBucket);
return copyRes.copyObjectResult().toString();

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

获取对象的法定保留配置

以下代码示例演示如何获取 S3 存储桶的法定保留配置。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
// 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;
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetObjectLegalHold](#)”中的。

锁定 Amazon S3 对象

以下代码示例演示了如何使用 Amazon S3 对象锁定功能。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

运行演示 Amazon S3 对象锁定功能的交互式场景。

```
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import java.io.BufferedWriter;
import java.io.IOException;
```

```
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.stream.Collectors;

/*
Before running this Java V2 code example, set up your development
environment, including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html

This Java example performs the following tasks:
  1. Create test Amazon Simple Storage Service (S3) buckets with different lock
  policies.
  2. Upload sample objects to each bucket.
  3. Set some Legal Hold and Retention Periods on objects and buckets.
  4. Investigate lock policies by viewing settings or attempting to delete or
  overwrite objects.
  5. Clean up objects and buckets.
*/
public class S3ObjectLockWorkflow {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    static String bucketName;
    static S3LockActions s3LockActions;
    private static final List<String> bucketNames = new ArrayList<>();
    private static final List<String> fileNames = new ArrayList<>();

    public static void main(String[] args) {
        // Get the current date and time to ensure bucket name is unique.
        LocalDateTime currentTime = LocalDateTime.now();

        // Format the date and time as a string.
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyyMMddHHmmss");
        String timeStamp = currentTime.format(formatter);

        s3LockActions = new S3LockActions();
        bucketName = "bucket"+timeStamp;
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
```

```
        System.out.println("Welcome to the Amazon Simple Storage Service (S3) Object  
Locking Workflow Scenario.");  
        System.out.println("Press Enter to continue...");  
        scanner.nextLine();  
        configurationSetup();  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);  
        setup();  
        System.out.println("Setup is complete. Press Enter to continue...");  
        scanner.nextLine();  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);  
        System.out.println("Lets present the user with choices.");  
        System.out.println("Press Enter to continue...");  
        scanner.nextLine();  
        demoActionChoices() ;  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);  
        System.out.println("Would you like to clean up the resources? (y/n)");  
        String delAns = scanner.nextLine().trim();  
        if (delAns.equalsIgnoreCase("y")) {  
            cleanup();  
            System.out.println("Clean up is complete.");  
        }  
  
        System.out.println("Press Enter to continue...");  
        scanner.nextLine();  
        System.out.println(DASHES);  
  
        System.out.println(DASHES);  
        System.out.println("Amazon S3 Object Locking Workflow is complete.");  
        System.out.println(DASHES);  
    }  
  
    // Present the user with the demo action choices.  
    public static void demoActionChoices() {  
        String[] choices = {  
            "List all files in buckets.",  
            "Attempt to delete a file.",  
            "Attempt to delete a file with retention period bypass.",  
            "Attempt to overwrite a file.",
```

```
        "View the object and bucket retention settings for a file.",
        "View the legal hold settings for a file.",
        "Finish the workflow."
    };

    int choice = 0;
    while (true) {
        System.out.println(DASHES);
        choice = getChoiceResponse("Explore the S3 locking features by selecting
one of the following choices:", choices);
        System.out.println(DASHES);
        System.out.println("You selected "+choices[choice]);
        switch (choice) {
            case 0 -> {
                s3LockActions.listBucketsAndObjects(bucketNames, true);
            }

            case 1 -> {
                System.out.println("Enter the number of the object to delete:");
                List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                String[] fileKeysArray = fileKeys.toArray(new String[0]);
                int fileChoice = getChoiceResponse(null, fileKeysArray);
                String objectKey = fileKeys.get(fileChoice);
                String bucketName = allFiles.get(fileChoice).getBucketName();
                String version = allFiles.get(fileChoice).getVersion();
                s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
false, version);
            }

            case 2 -> {
                System.out.println("Enter the number of the object to delete:");
                List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                String[] fileKeysArray = fileKeys.toArray(new String[0]);
                int fileChoice = getChoiceResponse(null, fileKeysArray);
                String objectKey = fileKeys.get(fileChoice);
                String bucketName = allFiles.get(fileChoice).getBucketName();
                String version = allFiles.get(fileChoice).getVersion();
            }
        }
    }
}
```



```
        s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
true, version);
    }

    case 3 -> {
        System.out.println("Enter the number of the object to
overwrite:");

        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
        List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
        String[] fileKeysArray = fileKeys.toArray(new String[0]);
        int fileChoice = getChoiceResponse(null, fileKeysArray);
        String objectKey = fileKeys.get(fileChoice);
        String bucketName = allFiles.get(fileChoice).getBucketName();

        // Attempt to overwrite the file.
        try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(objectKey))) {
            writer.write("This is a modified text.");

        } catch (IOException e) {
            e.printStackTrace();
        }
        s3LockActions.uploadFile(bucketName, objectKey, objectKey);
    }

    case 4 -> {
        System.out.println("Enter the number of the object to
overwrite:");

        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
        List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
        String[] fileKeysArray = fileKeys.toArray(new String[0]);
        int fileChoice = getChoiceResponse(null, fileKeysArray);
        String objectKey = fileKeys.get(fileChoice);
        String bucketName = allFiles.get(fileChoice).getBucketName();
        s3LockActions.getObjectRetention(bucketName, objectKey);
    }

    case 5 -> {
        System.out.println("Enter the number of the object to view:");
```

```

        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
        List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
        String[] fileKeysArray = fileKeys.toArray(new String[0]);
        int fileChoice = getChoiceResponse(null, fileKeysArray);
        String objectKey = fileKeys.get(fileChoice);
        String bucketName = allFiles.get(fileChoice).getBucketName();
        s3LockActions.getObjectLegalHold(bucketName, objectKey);
        s3LockActions.getBucketObjectLockConfiguration(bucketName);
    }

    case 6 -> {
        System.out.println("Exiting the workflow...");
        return;
    }

    default -> {
        System.out.println("Invalid choice. Please select again.");
    }
}
}

// Clean up the resources from the scenario.
private static void cleanup() {
    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, false);
    for (S3InfoObject fileInfo : allFiles) {
        String bucketName = fileInfo.getBucketName();
        String key = fileInfo.getKeyName();
        String version = fileInfo.getVersion();
        if (bucketName.contains("lock-enabled") ||
(bucketName.contains("retention-after-creation"))) {
            ObjectLockLegalHold legalHold =
s3LockActions.getObjectLegalHold(bucketName, key);
            if (legalHold != null) {
                String holdStatus = legalHold.status().name();
                System.out.println(holdStatus);
                if (holdStatus.compareTo("ON") == 0) {
                    s3LockActions.modifyObjectLegalHold(bucketName, key, false);
                }
            }
        }
    }
    // Check for a retention period.

```

```
        ObjectLockRetention retention =
s3LockActions.getObjectRetention(bucketName, key);
        boolean hasRetentionPeriod ;
        hasRetentionPeriod = retention != null;
        s3LockActions.deleteObjectFromBucket(bucketName,
key,hasRetentionPeriod, version);

    } else {
        System.out.println(bucketName + " objects do not have a legal lock");
        s3LockActions.deleteObjectFromBucket(bucketName, key,false,
version);
    }
}

// Delete the buckets.
System.out.println("Delete "+bucketName);
for (String bucket : bucketNames){
    s3LockActions.deleteBucketByName(bucket);
}
}

private static void setup() {
    Scanner scanner = new Scanner(System.in);
    System.out.println("""
        For this workflow, we will use the AWS SDK for Java to create
several S3
        buckets and files to demonstrate working with S3 locking features.
        """);

    System.out.println("S3 buckets can be created either with or without object
lock enabled.");
    System.out.println("Press Enter to continue...");
    scanner.nextLine();

    // Create three S3 buckets.
    s3LockActions.createBucketWithLockOptions(false, bucketNames.get(0));
    s3LockActions.createBucketWithLockOptions(true, bucketNames.get(1));
    s3LockActions.createBucketWithLockOptions(false, bucketNames.get(2));
    System.out.println("Press Enter to continue.");
    scanner.nextLine();

    System.out.println("Bucket "+bucketNames.get(2) +" will be configured to use
object locking with a default retention period.");
    s3LockActions.modifyBucketDefaultRetention(bucketNames.get(2));
```

```
System.out.println("Press Enter to continue.");
scanner.nextLine();

System.out.println("Object lock policies can also be added to existing
buckets. For this example, we will use "+bucketNames.get(1));
s3LockActions.enableObjectLockOnBucket(bucketNames.get(1));
System.out.println("Press Enter to continue.");
scanner.nextLine();

// Upload some files to the buckets.
System.out.println("Now let's add some test files:");
String fileName = "exampleFile.txt";
int fileCount = 2;
try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(fileName))) {
    writer.write("This is a sample file for uploading to a bucket.");

} catch (IOException e) {
    e.printStackTrace();
}

for (String bucketName : bucketNames){
    for (int i = 0; i < fileCount; i++) {
        // Get the file name without extension.
        String fileNameWithoutExtension =
java.nio.file.Paths.get(fileName).getFileName().toString();
        int extensionIndex = fileNameWithoutExtension.lastIndexOf('.');
        if (extensionIndex > 0) {
            fileNameWithoutExtension = fileNameWithoutExtension.substring(0,
extensionIndex);
        }

        // Create the numbered file names.
        String numberedFileName = fileNameWithoutExtension + i +
getFileExtension(fileName);
        fileNames.add(numberedFileName);
        s3LockActions.uploadFile(bucketName, numberedFileName, fileName);
    }
}

String question = null;
System.out.print("Press Enter to continue...");
scanner.nextLine();
```

```

        System.out.println("Now we can set some object lock policies on individual
files:");
        for (String bucketName : bucketNames) {
            for (int i = 0; i < fileNames.size(); i++){

                // No modifications to the objects in the first bucket.
                if (!bucketName.equals(bucketNames.get(0))) {
                    String exampleFileName = fileNames.get(i);
                    switch (i) {
                        case 0 -> {
                            question = "Would you like to add a legal hold to " +
exampleFileName + " in " + bucketName + " (y/n)?";
                            System.out.println(question);
                            String ans = scanner.nextLine().trim();
                            if (ans.equalsIgnoreCase("y")) {
                                System.out.println("**** You have selected to put a
legal hold " + exampleFileName);

                                    // Set a legal hold.
                                    s3LockActions.modifyObjectLegalHold(bucketName,
exampleFileName, true);
                                }
                            }
                        case 1 -> {
                            """"
                                Would you like to add a 1 day Governance retention
period to %s in %s (y/n)?

                                    Reminder: Only a user with the
s3:BypassGovernanceRetention permission will be able to delete this file or its
bucket until the retention period has expired.
                                """".formatted(exampleFileName, bucketName);
                            System.out.println(question);
                            String ans2 = scanner.nextLine().trim();
                            if (ans2.equalsIgnoreCase("y")) {

                                s3LockActions.modifyObjectRetentionPeriod(bucketName, exampleFileName);
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```
// Get file extension.
private static String getFileExtension(String fileName) {
    int dotIndex = fileName.lastIndexOf('.');
    if (dotIndex > 0) {
        return fileName.substring(dotIndex);
    }
    return "";
}

public static void configurationSetup() {
    String noLockBucketName = bucketName + "-no-lock";
    String lockEnabledBucketName = bucketName + "-lock-enabled";
    String retentionAfterCreationBucketName = bucketName + "-retention-after-creation";
    bucketNames.add(noLockBucketName);
    bucketNames.add(lockEnabledBucketName);
    bucketNames.add(retentionAfterCreationBucketName);
}

public static int getChoiceResponse(String question, String[] choices) {
    Scanner scanner = new Scanner(System.in);
    if (question != null) {
        System.out.println(question);
        for (int i = 0; i < choices.length; i++) {
            System.out.println("\t" + (i + 1) + ". " + choices[i]);
        }
    }

    int choiceNumber = 0;
    while (choiceNumber < 1 || choiceNumber > choices.length) {
        String choice = scanner.nextLine();
        try {
            choiceNumber = Integer.parseInt(choice);
        } catch (NumberFormatException e) {
            System.out.println("Invalid choice. Please enter a valid number.");
        }
    }

    return choiceNumber - 1;
}
}
```

S3 函数的包装器类。

```
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 retention = DefaultRetention.builder()
        .days(1)
        .mode(ObjectLockRetentionMode.GOVERNANCE)
        .build();

    ObjectLockRule lockRule = ObjectLockRule.builder()
        .defaultRetention(retention)
        .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.getObjectLockConfiguration().getObjectLockEnabled());
    System.out.println("\tRule: "+
response.getObjectLockConfiguration().rule().defaultRetention());
}
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)
 - [PutObjectLockConfiguration](#)
 - [PutObjectRetention](#)

解析 URIs

以下代码示例说明如何解析 Amazon S3 URIs 以提取存储桶名称和对象密钥等重要组件。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用 S3Uri 类解析亚马逊 [S3](#)。

```
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);
    }
}
```

执行分段上传

以下代码示例显示如何执行 Amazon S3 对象的分段上传。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

代码示例使用以下导入。

```
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;
```

当内容大小超过阈值时，使用[AWS CRT 基于的 S3 客户端](#)之上的 S3 [传输管理器](#)透明地执行分段上传。默认阈值大小为 8 MB。

```
public void multipartUploadWithTransferManager(String filePath) {
    S3TransferManager transferManager = S3TransferManager.create();
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b
            .bucket(bucketName)
            .key(key))
        .source(Paths.get(filePath))
        .build();
    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
    fileUpload.completionFuture().join();
    transferManager.close();
}
```

使用 [S3Client API](#) 执行分段上传。

```
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()));
}
```

使用启用了多部分支持的 [S3 AsyncClient API](#) 来执行分段上传。

```
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.");
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [CompleteMultipartUpload](#)
 - [CreateMultipartUpload](#)
 - [UploadPart](#)

处理 S3 事件通知

以下代码示例显示了如何以面向对象的方式处理 S3 事件通知。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

此示例说明如何使用 Amazon 处理 S3 通知事件SQS。

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

```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [DeleteMessageBatch](#)
 - [GetQueueAttributes](#)

- [PutBucketNotificationConfiguration](#)
- [ReceiveMessage](#)

将事件通知发送至 EventBridge

以下代码示例演示如何使存储桶能够将 S3 事件通知发送到亚马逊主题 EventBridge 和亚马逊队列，以及如何将通知发送到亚马逊 SNS 主题和亚马逊 SQS 队列。

SDK 适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/** 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;
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [PutBucketNotificationConfiguration](#)
 - [PutRule](#)
 - [PutTargets](#)

跟踪上传和下载操作

下面的代码示例展示了如何跟踪 Amazon S3 对象的上传或下载。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

跟踪文件上传进度。

```
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!

*/
}
```

跟踪文件下载进度。

```
public void trackDownloadFile(S3TransferManager transferManager, String
bucketName,
                               String key, String downloadedFilePath) {
  DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
    .getObjectRequest(b -> b.bucket(bucketName).key(key))
    .addTransferListener(LoggingTransferListener.create()) // Add
listener.
    .destination(Paths.get(downloadedFilePath))
```

```

        .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!

        */
    }

```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [GetObject](#)
 - [PutObject](#)

将目录上传到存储桶

以下代码示例显示如何以递归方式将本地目录上传到 Amazon Simple Storage Service (Amazon S3) 桶。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用 [S3 TransferManager](#) 上传本地目录。查看 [完整文件](#) 并 [进行测试](#)。

```
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();
    }
```

```
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [UploadDirectory](#)”中的。

上传或下载大文件

下面的代码示例展示了如何向 Amazon S3 上传大文件或从 Amazon S3 下载大文件。

有关更多信息，请参阅[使用分段上传操作上传对象](#)。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

调用使用 S3 将文件传入和传出 S3 存储桶的函数TransferManager。

```
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();
}
```

上传整个本地目录。

```
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();
}
```

上传单个文件。

```
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();
}
```

上传未知大小的流

下面的代码示例演示了如何将未知大小的流上传到 Amazon S3 对象。

SDK适用于 Java 2.x

 Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

使用 [AWS CRT 基于的 S3 客户端](#)。

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

/**
 * @param s3CrtAsyncClient - To upload content from a stream of unknown size,
 * use the AWS CRT-based S3 client. For more information, see
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/crt-based-s3-client.html.
 * @param bucketName - The name of the bucket.
 * @param key - The name of the object.
 * @return software.amazon.awssdk.services.s3.model.PutObjectResponse - Returns
 * metadata pertaining to the put object operation.
 */
public PutObjectResponse putObjectFromStream(S3AsyncClient s3CrtAsyncClient,
String bucketName, String key) {

    BlockingInputStreamAsyncRequestBody body =
        AsyncRequestBody.forBlockingInputStream(null); // 'null' indicates a
stream will be provided later.

    CompletableFuture<PutObjectResponse> responseFuture =
        s3CrtAsyncClient.putObject(r -> r.bucket(bucketName).key(key),
body);

    // AsyncExampleUtils.randomString() returns a random string up to 100
characters.
    String randomString = AsyncExampleUtils.randomString();
    logger.info("random string to upload: {}: length={}", randomString,
randomString.length());

    // Provide the stream of data to be uploaded.
    body.writeInputStream(new ByteArrayInputStream(randomString.getBytes()));
}
```

```
        PutObjectResponse response = responseFuture.join(); // Wait for the
response.
        logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        return response;
    }
}
```

使用 [Amazon S3 Transfer Manager](#)。

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedUpload;
import software.amazon.awssdk.transfer.s3.model.Upload;

import java.io.ByteArrayInputStream;
import java.util.UUID;

/**
 * @param transferManager - To upload content from a stream of unknown size, use
the S3TransferManager based on the AWS CRT-based S3 client.
 *
 * For more information, see https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/transfer-manager.html.
 * @param bucketName - The name of the bucket.
 * @param key - The name of the object.
 * @return - software.amazon.awssdk.transfer.s3.model.CompletedUpload - The
result of the completed upload.
 */
public CompletedUpload uploadStream(S3TransferManager transferManager, String
bucketName, String key) {

    BlockingInputStreamAsyncRequestBody body =
        AsyncRequestBody.forBlockingInputStream(null); // 'null' indicates a
stream will be provided later.

    Upload upload = transferManager.upload(builder -> builder
        .requestBody(body)
```

```
        .putObjectRequest(req -> req.bucket(bucketName).key(key))
        .build());

    // AsyncExampleUtils.randomString() returns a random string up to 100
    characters.
    String randomString = AsyncExampleUtils.randomString();
    logger.info("random string to upload: {}: length={}", randomString,
randomString.length());

    // Provide the stream of data to be uploaded.
    body.writeInputStream(new ByteArrayInputStream(randomString.getBytes()));

    return upload.completionFuture().join();
}
}
```

使用校验和

以下代码示例显示如何对 Amazon S3 对象使用校验和。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

代码示例使用以下导入的子集。

```
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;
```

在构建 [PutObjectRequest](#) 时为 `putObject` 方法指定校验和算法。

```
public void putObjectWithChecksum() {
    s3Client.putObject(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumAlgorithm(ChecksumAlgorithm.CRC32),
        RequestBody.fromString("This is a test"));
}
```

在构建时验证该 [getObject](#) 方法的校验和。 [GetObjectRequest](#)

```
public GetObjectResponse getObjectWithChecksum() {
    return s3Client.getObject(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumMode(ChecksumMode.ENABLED))
```

```
        .response();  
    }
```

在[构建 PutObjectRequest](#) 时为 putObject 方法预先计算校验和。

```
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)));  
}
```

当内容大小超过阈值时，使用[AWS CRT 基于的 S3 客户端](#)之上的 S3 [传输管理器](#)透明地执行分段上传。默认阈值大小为 8 MB。

您可以指定要使用的校验和算法 SDK。默认情况下，SDK 使用 CRC32 算法。

```
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();  
}
```

使用 [S3Client API](#) 或 (S3 AsyncClient API) 执行分段上传。如果指定其他校验和，则您必须指定在启动上传时使用的算法。您还必须为每个分段请求指定算法，并提供每个分段在上传后计算得出的校验和。

```
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()));
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [CompleteMultipartUpload](#)
 - [CreateMultipartUpload](#)
 - [UploadPart](#)

无服务器示例

通过 Amazon S3 触发器调用 Lambda 函数

以下代码示例展示了如何实现一个 Lambda 函数，该函数接收通过将对象上传到 S3 桶而触发的事件。该函数从事件参数中检索 S3 存储桶名称和对象密钥，并调用 Amazon S3 API 来检索和记录对象的内容类型。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

使用 Java 将 S3 事件与 Lambda 结合使用。

```
// 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适用于 Java 2.x 的 Amazon S3 控制示例

以下代码示例向您展示了如何使用 AWS SDK for Java 2.x 与 Amazon S3 Control 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

开始使用

你好 Amazon S3 Control

以下代码示例显示了如何开始使用“Amazon S3 控件”

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import
    software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
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.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;

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(); // Wait for completion

        } 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>
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)

.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .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);
        });
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListJobsPaginator](#)”中的。

主题

- [操作](#)
- [场景](#)

操作

CreateJob

以下代码示例演示如何使用 CreateJob。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

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

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateJob](#)”中的。

DeleteJobTagging

以下代码示例演示如何使用 DeleteJobTagging。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

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

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteJobTagging](#)”中的。

DescribeJob

以下代码示例演示如何使用 DescribeJob。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。


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

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeJob](#)”中的。

GetJobTagging

以下代码示例演示如何使用 GetJobTagging。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

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

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetJobTagging](#)”中的。

PutJobTagging

以下代码示例演示如何使用 PutJobTagging。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

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

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [PutJobTagging](#)”中的。

UpdateJobPriority

以下代码示例演示如何使用 UpdateJobPriority。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**
 * 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;
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [UpdateJobPriority](#)”中的。

UpdateJobStatus

以下代码示例演示如何使用 UpdateJobStatus。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

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

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [UpdateJobStatus](#)”中的。

场景

学习核心运营

以下代码示例展示了如何学习“Amazon S3 控制”的核心操作。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

学习核心操作。

```
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 = "x-" + UUID.randomUUID();
    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.");
    }
}
}
```

封装操作的操作类。

```
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)

.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .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))
            .retryPolicy(RetryPolicy.builder()
                .numRetries(3)
                .build())
            .build();
    }
}
```

```
        .build())
        .build();

    s3AsyncClient = S3AsyncClient.builder()
        .region(Region.US_EAST_1)
        .httpClient(httpClient)
        .overrideConfiguration(overrideConfig)

.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
        .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();
}
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [CreateJob](#)
 - [DeleteJobTagging](#)

- [DescribeJob](#)
- [GetJobTagging](#)
- [ListJobs](#)
- [PutJobTagging](#)
- [UpdateJobPriority](#)
- [UpdateJobStatus](#)

用于 Java 2.x SDK 的 S3 Glacier 示例

以下代码示例向您展示了如何在 S3 Glacier 中使用来执行操作和实现常见场景。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

CreateVault

以下代码示例演示如何使用 CreateVault。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateVault](#)”中的。

DeleteArchive

以下代码示例演示如何使用 DeleteArchive。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

```
    }  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteArchive](#)”中的。

DeleteVault

以下代码示例演示如何使用 DeleteVault。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteVault](#)”中的。

InitiateJob

以下代码示例演示如何使用 InitiateJob。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

检索文件库库存。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [InitiateJob](#)”中的。

ListVaults

以下代码示例演示如何使用 ListVaults。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListVaults](#)”中的。

UploadArchive

以下代码示例演示如何使用 UploadArchive。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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();
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [UploadArchive](#)”中的。

SageMaker 用SDK于 Java 2.x 的示例

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 SageMaker。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

开始使用

你好 SageMaker

以下代码示例展示了如何开始使用 SageMaker。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * 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);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListNotebookInstances](#)”中的。

主题

- [操作](#)


- [场景](#)

操作

CreatePipeline

以下代码示例演示如何使用 CreatePipeline。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

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

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreatePipeline](#)”中的。

DeletePipeline

以下代码示例演示如何使用 DeletePipeline。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
// 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);
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeletePipeline](#)”中的。

DescribePipelineExecution

以下代码示例演示如何使用 DescribePipelineExecution。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
// 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);
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribePipelineExecution](#)”中的。

StartPipelineExecution

以下代码示例演示如何使用 StartPipelineExecution。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

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

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartPipelineExecution](#)”中的。

场景

开始使用地理空间作业和管道

以下代码示例演示了操作流程：

- 为管道设置资源。
- 设置用于执行地理空间作业的管道。
- 启动管道执行。
- 监控执行的状态。
- 查看管道的输出。
- 清理资源。

有关更多信息，请参阅在 [Community.aws AWS SDKs 上使用创建和运行 SageMaker 管道](#)。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 =
sqscClient.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;
}
}

```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [CreatePipeline](#)
 - [DeletePipeline](#)
 - [DescribePipelineExecution](#)
 - [StartPipelineExecution](#)

- [UpdatePipeline](#)

用SDK于 Java 2.x 的 Secrets Manager 示例

以下代码示例向您展示了如何使用 with Secrets Manager 来执行操作和实现常见场景。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

GetSecretValue

以下代码示例演示如何使用 GetSecretValue。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetSecretValue](#)”中的。

使用SDK适用于 Java 2.x 的亚马逊SES示例

以下代码示例向您展示如何在 Amazon 中使用来执行操作和实现常见场景SES。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

ListIdentities

以下代码示例演示如何使用 ListIdentities。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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();

        listSESIIdentities(client);
    }

    public static void listSESIIdentities(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);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListIdentities](#)”中的。

ListTemplates

以下代码示例演示如何使用 ListTemplates。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListTemplates](#)”中的。

SendEmail

以下代码示例演示如何使用 SendEmail。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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");
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [SendEmail](#)”中的。

SendTemplatedEmail

以下代码示例演示如何使用 SendTemplatedEmail。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [SendTemplatedEmail](#)”中的。

使用SDK适用于 Java 2.x 的亚马逊 SES API v2 示例

以下代码示例向您展示了如何在 Amazon SES API v2 中使用来执行操作和实现常见场景。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)
- [场景](#)

操作

CreateContact

以下代码示例演示如何使用 CreateContact。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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;
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateContact](#)”中的。

CreateContactList

以下代码示例演示如何使用 CreateContactList。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。


```
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;
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateContactList](#)”中的。

CreateEmailIdentity

以下代码示例演示如何使用 CreateEmailIdentity。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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;
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateEmailIdentity](#)”中的。

CreateEmailTemplate

以下代码示例演示如何使用 CreateEmailTemplate。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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;
}
```

```
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateEmailTemplate](#)”中的。

DeleteContactList

以下代码示例演示如何使用 DeleteContactList。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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();
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteContactList](#)”中的。

DeleteEmailIdentity

以下代码示例演示如何使用 DeleteEmailIdentity。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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.");
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteEmailIdentity](#)”中的。

DeleteEmailTemplate

以下代码示例演示如何使用 DeleteEmailTemplate。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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();
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteEmailTemplate](#)”中的。

ListContacts

以下代码示例演示如何使用 ListContacts。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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;
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListContacts](#)”中的。

SendEmail

以下代码示例演示如何使用 SendEmail。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

发送邮件。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sesv2.model.Body;
import software.amazon.awssdk.services.sesv2.model.Content;
import software.amazon.awssdk.services.sesv2.model.Destination;
import software.amazon.awssdk.services.sesv2.model.EmailContent;
import software.amazon.awssdk.services.sesv2.model.Message;
```

```
import software.amazon.awssdk.services.sesv2.model.SendEmailRequest;
import software.amazon.awssdk.services.sesv2.model.SesV2Exception;
import software.amazon.awssdk.services.sesv2.SesV2Client;

/**
 * Before running this AWS SDK for Java (v2) example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class SendEmail {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <sender> <recipient> <subject>\s

            Where:
                sender - An email address that represents the
sender.\s
                recipient - An email address that represents the
recipient.\s
                subject - The subject line.\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];

        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
            .region(region)
            .build();

        // The HTML body of the email.
    }
}
```

```
String bodyHTML = "<html>" + "<head></head>" + "<body>" +
"<h1>Hello!</h1>"
                + "<p> See the list of customers.</p>" + "</body>" +
"</html>";

    send(sesv2Client, sender, recipient, subject, bodyHTML);
}

public static void send(SesV2Client client,
    String sender,
    String recipient,
    String subject,
    String bodyHTML) {

    Destination destination = Destination.builder()
        .toAddresses(recipient)
        .build();

    Content content = Content.builder()
        .data(bodyHTML)
        .build();

    Content sub = Content.builder()
        .data(subject)
        .build();

    Body body = Body.builder()
        .html(content)
        .build();

    Message msg = Message.builder()
        .subject(sub)
        .body(body)
        .build();

    EmailContent emailContent = EmailContent.builder()
        .simple(msg)
        .build();

    SendEmailRequest emailRequest = SendEmailRequest.builder()
        .destination(destination)
        .content(emailContent)
        .fromEmailAddress(sender)
        .build();
```

```
        try {
            System.out.println("Attempting to send an email through
Amazon SES "
                               + "using the AWS SDK for Java...");
            client.sendEmail(emailRequest);
            System.out.println("email was sent");
        } catch (SesV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

使用模板发送消息。

```
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());
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [SendEmail](#)”中的。

场景

时事通讯工作流程

以下代码示例显示了如何运行 Amazon SES API v2 新闻简报工作流程。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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();
}
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。

- [CreateContact](#)
- [CreateContactList](#)
- [CreateEmailIdentity](#)
- [CreateEmailTemplate](#)
- [DeleteContactList](#)
- [DeleteEmailIdentity](#)
- [DeleteEmailTemplate](#)
- [ListContacts](#)
- [SendEmail。simple](#)
- [SendEmail。模板](#)

使用SDK适用于 Java 2.x 的亚马逊SNS示例

以下代码示例向您展示如何在 Amazon 中使用来执行操作和实现常见场景SNS。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

开始使用

你好 Amazon SNS

以下代码示例展示了如何开始使用 Amazon SNS。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListTopics](#)”中的。

主题

- [操作](#)
- [场景](#)
- [无服务器示例](#)

操作

CheckIfPhoneNumberIsOptedOut

以下代码示例演示如何使用 CheckIfPhoneNumberIsOptedOut。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CheckIfPhoneNumberIsOptedOut](#)”中的。

ConfirmSubscription

以下代码示例演示如何使用 ConfirmSubscription。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ConfirmSubscription](#)”中的。

CreateTopic

以下代码示例演示如何使用 CreateTopic。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";
    }
}
```

```
}  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateTopic](#)”中的。

DeleteTopic

以下代码示例演示如何使用 DeleteTopic。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteTopic](#)”中的。

GetSMSAttributes

以下代码示例演示如何使用 GetSMSAttributes。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 GetSMSAttributes {
    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");
}
}
```

- 有关API详细信息，请参阅etSMSAttributes 《AWS SDK for Java 2.x API参考资料》中的 [G](#)。

GetTopicAttributes

以下代码示例演示如何使用 GetTopicAttributes。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetTopicAttributes](#)”中的。

ListPhoneNumbersOptedOut

以下代码示例演示如何使用 ListPhoneNumbersOptedOut。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListPhoneNumbersOptedOut](#)”中的。

ListSubscriptions

以下代码示例演示如何使用 ListSubscriptions。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);  
        }  
    }  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListSubscriptions](#)”中的。

ListTopics

以下代码示例演示如何使用 ListTopics。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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\n"
                + result.topics());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListTopics](#)”中的。

Publish

以下代码示例演示如何使用 Publish。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅在AWS SDK for Java 2.x API参考中[发布](#)。

SetSMSAttributes

以下代码示例演示如何使用 SetSMSAttributes。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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();
        setSMSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSMSAttributes(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);
        }
    }
}
```

- 有关API详细信息，请参阅setSMSAttributes 《AWS SDK for Java 2.x API参考资料》中的 [S](#)。

SetSubscriptionAttributes

以下代码示例演示如何使用 SetSubscriptionAttributes。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [SetSubscriptionAttributes](#)”中的。

SetTopicAttributes

以下代码示例演示如何使用 SetTopicAttributes。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [SetTopicAttributes](#)”中的。

Subscribe

以下代码示例演示如何使用 Subscribe。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

通过电子邮件地址订阅主题。

```
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);
    }
}
}
}

```

为HTTP终端节点订阅主题。

```

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);
    }
}
}
```

为主题订阅 Lambda 函数。

```
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 "";
}
}
```

- 有关API详细信息，请参阅AWS SDK for Java 2.x API参考资料中的[订阅](#)。

TagResource

以下代码示例演示如何使用 TagResource。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [TagResource](#)”中的。

Unsubscribe

以下代码示例演示如何使用 Unsubscribe。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考资料”中的[“取消订阅”](#)。

场景

为推送通知创建平台终端节点

以下代码示例显示如何为 Amazon SNS 推送通知创建平台终端节点。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointRequest;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
 */

public class RegistrationExample {
    public static void main(String[] args) {
        final String usage = ""

                Usage:      <token> <platformApplicationArn>

                Where:
                    token - The name of the FIFO topic.\s
                    platformApplicationArn - The ARN value of platform application.
                You can get this value from the AWS Management Console.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String token = args[0];
        String platformApplicationArn = args[1];
        SnsClient snsClient = SnsClient.builder()
```

```
        .region(Region.US_EAST_1)
        .build();

    createEndpoint(snsClient, token, platformApplicationArn);
}

public static void createEndpoint(SnsClient snsClient, String token, String
platformApplicationArn) {
    System.out.println("Creating platform endpoint with token " + token);
    try {
        CreatePlatformEndpointRequest endpointRequest =
CreatePlatformEndpointRequest.builder()
            .token(token)
            .platformApplicationArn(platformApplicationArn)
            .build();

        CreatePlatformEndpointResponse response =
snsClient.createPlatformEndpoint(endpointRequest);
        System.out.println("The ARN of the endpoint is " +
response.endpointArn());
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

创建并发布到FIFO主题

以下代码示例展示了如何创建和发布到 A FIFO mazon SNS 主题。

SDK适用于 Java 2.x

Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

此示例

- 创建一个 Amazon SNS FIFO 主题、两个亚马逊SQSFIFO队列和一个标准队列。
- 将队列订阅到主题，发布一条消息到主题。

该[测试](#)验证每个队列是否收到消息。[完整的示例](#)还显示了添加访问策略，并在最后删除了资源。

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {

        final String usage = "\n" +
            "Usage: " +
            "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
<analyticsQueueName>\n\n" +
            "Where:\n" +
            "    fifoTopicName - The name of the FIFO topic that you want to
create. \n\n" +
            "    wholesaleQueueARN - The name of a SQS FIFO queue that will be
created for the wholesale consumer. \n\n"
            +
            "    retailQueueARN - The name of a SQS FIFO queue that will created
for the retail consumer. \n\n" +
            "    analyticsQueueARN - The name of a SQS standard queue that will
be created for the analytics consumer. \n\n";
        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        final String fifoTopicName = args[0];
        final String wholeSaleQueueName = args[1];
        final String retailQueueName = args[2];
        final String analyticsQueueName = args[3];

        // For convenience, the QueueData class holds metadata about a queue: ARN,
URL,
        // name and type.
        List<QueueData> queues = List.of(
            new QueueData(wholeSaleQueueName, QueueType.FIFO),
            new QueueData(retailQueueName, QueueType.FIFO),
            new QueueData(analyticsQueueName, QueueType.Standard));
```

```
// Create queues.
createQueues(queues);

// Create a topic.
String topicARN = createFIFOTopic(fifoTopicName);

// Subscribe each queue to the topic.
subscribeQueues(queues, topicARN);

// Allow the newly created topic to send messages to the queues.
addAccessPolicyToQueuesFINAL(queues, topicARN);

// Publish a sample price update message with payload.
publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

// Clean up resources.
deleteSubscriptions(queues);
deleteQueues(queues);
deleteTopic(topicARN);
}

public static String createFIFOTopic(String topicName) {
    try {
        // Create a FIFO topic by using the SNS service client.
        Map<String, String> topicAttributes = Map.of(
            "FifoTopic", "true",
            "ContentBasedDeduplication", "false");

        CreateTopicRequest topicRequest = CreateTopicRequest.builder()
            .name(topicName)
            .attributes(topicAttributes)
            .build();

        CreateTopicResponse response = snsClient.createTopic(topicRequest);
        String topicArn = response.topicArn();
        System.out.println("The topic ARN is" + topicArn);

        return topicArn;
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```



```
        return "";
    }

    public static void subscribeQueues(List<QueueData> queues, String topicARN) {
        queues.forEach(queue -> {
            SubscribeRequest subscribeRequest = SubscribeRequest.builder()
                .topicArn(topicARN)
                .endpoint(queue.queueARN)
                .protocol("sqs")
                .build();

            // Subscribe to the endpoint by using the SNS service client.
            // Only Amazon SQS queues can receive notifications from an Amazon SNS
            FIFO
            // topic.
            SubscribeResponse subscribeResponse =
            snsClient.subscribe(subscribeRequest);
            System.out.println("The queue [" + queue.queueARN + "] subscribed to the
            topic [" + topicARN + "]);
            queue.subscriptionARN = subscribeResponse.subscriptionArn();
        });
    }

    public static void publishPriceUpdate(String topicArn, String payload, String
    groupId) {

        try {
            // Create and publish a message that updates the wholesale price.
            String subject = "Price Update";
            String dedupId = UUID.randomUUID().toString();
            String attributeName = "business";
            String attributeValue = "wholesale";

            MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
                .dataType("String")
                .stringValue(attributeValue)
                .build();

            Map<String, MessageAttributeValue> attributes = new HashMap<>();
            attributes.put(attributeName, msgAttValue);
            PublishRequest pubRequest = PublishRequest.builder()
                .topicArn(topicArn)
                .subject(subject)
                .message(payload)
```

```
        .messageGroupId(groupId)
        .messageDeduplicationId(dedupId)
        .messageAttributes(attributes)
        .build();

    final PublishResponse response = snsClient.publish(pubRequest);
    System.out.println(response.messageId());
    System.out.println(response.sequenceNumber());
    System.out.println("Message was published to " + topicArn);

} catch (SnsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [CreateTopic](#)
 - [Publish](#)
 - [订阅](#)

向主题发布SMS消息

以下代码示例展示了如何：

- 创建 Amazon SNS 主题。
- 使用手机号码订阅主题。
- 向主题发布SMS消息，以便所有订阅的电话号码都能同时收到消息。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

创建一个主题并返回它ARN。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <topicName>

            Where:
                topicName - The name of the topic to create (for example,
mytopic).

            "";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
```

```
    CreateTopicResponse result;
    try {
        CreateTopicRequest request = CreateTopicRequest.builder()
            .name(topicName)
            .build();

        result = snsClient.createTopic(request);
        return result.topicArn();

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

为终端节点订阅主题。

```
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);
    }
}
}
}

```

设置消息的属性，例如发件人的 ID、最高价格及其类型。消息属性是可选的。

```
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);
        }
    }
}
```

```
    }  
  }  
}
```

向主题发布消息。消息将会发送到每个订阅者。

```
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);
        }
    }
}
```

发布一条SMS短信

以下代码示例展示了如何使用 Amazon 发布SMS消息SNS。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关API详细信息，请参阅在AWS SDK for Java 2.x API参考中[发布](#)。

将消息发布到队列

以下代码示例展示了如何：

- 创建主题（FIFO或非-FIFO）。
- 针对主题订阅多个队列，并提供应用筛选条件的选项。
- 将消息发布到主题。
- 轮询队列中是否有收到的消息。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
package com.example.sns;

import
    software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
```

```
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.MessageAttributeValue;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import com.google.gson.Gson;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonPrimitive;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java example performs these tasks:
```

```
*
* 1. Gives the user three options to choose from.
* 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
* 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
* 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
* 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
* 6. Subscribes to the SQS queue.
* 7. Publishes a message to the topic.
* 8. Displays the messages.
* 9. Deletes the received message.
* 10. Unsubscribes from the topic.
* 11. Deletes the SNS topic.
*/
public class SNSWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:\n" +
            "    <fifoQueueARN>\n\n" +
            "Where:\n" +
            "    accountId - Your AWS account Id value.";

        // if (args.length != 1) {
        // System.out.println(usage);
        // System.exit(1);
        // }

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)

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

        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_EAST_1)

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

        Scanner in = new Scanner(System.in);
        String accountId = "814548047983";
        String useFIFO;
        String duplication = "n";
```

```
String topicName;
String deduplicationID = null;
String groupId = null;

String topicArn;
String sqsQueueName;
String sqsQueueUrl;
String sqsQueueArn;
String subscriptionArn;
boolean selectFIFO = false;

String message;
List<Message> messageList;
List<String> filterList = new ArrayList<>();
String msgAttValue = "";

System.out.println(DASHES);
System.out.println("Welcome to messaging with topics and queues.");
System.out.println("In this workflow, you will create an SNS topic and
subscribe an SQS queue to the topic.\n" +
    "You can select from several options for configuring the topic and
the subscriptions for the queue.\n" +
    "You can then post to the topic and see the results in the queue.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("SNS topics can be configured as FIFO (First-In-First-
Out).\n" +
    "FIFO topics deliver messages in order and support deduplication and
message filtering.\n" +
    "Would you like to work with FIFO topics? (y/n)");
useFIFO = in.nextLine();
if (useFIFO.compareTo("y") == 0) {
    selectFIFO = true;
    System.out.println("You have selected FIFO");
    System.out.println(" Because you have chosen a FIFO topic, deduplication
is supported.\n" +
        "          Deduplication IDs are either set in the message or
automatically generated from content using a hash function.\n"
        +
        "          If a message is successfully published to an SNS FIFO
topic, any message published and determined to have the same deduplication ID,\n"
        +
```

```
        "        within the five-minute deduplication interval, is
accepted but not delivered.\n" +
        "        For more information about deduplication, see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.");

    System.out.println(
        "Would you like to use content-based deduplication instead of
entering a deduplication ID? (y/n)");
    duplication = in.nextLine();
    if (duplication.compareTo("y") == 0) {
        System.out.println("Please enter a group id value");
        groupId = in.nextLine();
    } else {
        System.out.println("Please enter deduplication Id value");
        deduplicationID = in.nextLine();
        System.out.println("Please enter a group id value");
        groupId = in.nextLine();
    }
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Create a topic.");
System.out.println("Enter a name for your SNS topic.");
topicName = in.nextLine();
if (selectFIFO) {
    System.out.println("Because you have selected a FIFO topic, '.fifo' must
be appended to the topic name.");
    topicName = topicName + ".fifo";
    System.out.println("The name of the topic is " + topicName);
    topicArn = createFIFO(snsClient, topicName, duplication);
    System.out.println("The ARN of the FIFO topic is " + topicArn);

} else {
    System.out.println("The name of the topic is " + topicName);
    topicArn = createSNSTopic(snsClient, topicName);
    System.out.println("The ARN of the non-FIFO topic is " + topicArn);

}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Create an SQS queue.");
System.out.println("Enter a name for your SQS queue.");
```

```

sqsQueueName = in.nextLine();
if (selectFIFO) {
    sqsQueueName = sqsQueueName + ".fifo";
}
sqsQueueUrl = createQueue(sqsClient, sqsQueueName, selectFIFO);
System.out.println("The queue URL is " + sqsQueueUrl);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Get the SQS queue ARN attribute.");
sqsQueueArn = getSQSQueueAttrs(sqsClient, sqsQueueUrl);
System.out.println("The ARN of the new queue is " + sqsQueueArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Attach an IAM policy to the queue.");

// Define the policy to use. Make sure that you change the REGION if you are
// running this code
// in a different region.
String policy = "{\n" +
    "    \"Statement\": [\n" +
    "        {\n" +
    "            \"Effect\": \"Allow\",\n" +
    "            \"Principal\": {\n" +
    "                \"Service\": \"sns.amazonaws.com\"\n" +
    "            },\n" +
    "            \"Action\": \"sqs:SendMessage\",\n" +
    "            \"Resource\": \"arn:aws:sqs:us-east-1:" +
accountId + ":" + sqsQueueName + "\",\n" +
    "                \"Condition\": {\n" +
    "                    \"ArnEquals\": {\n" +
    "                        \"aws:SourceArn\": \"arn:aws:sns:us-east-1:" +
accountId + ":" + topicName + "\"\n" +
    "                    }\n" +
    "                }\n" +
    "            }\n" +
    "        ]\n" +
    "    }";

setQueueAttr(sqsClient, sqsQueueUrl, policy);
System.out.println(DASHES);

System.out.println(DASHES);

```

```
System.out.println("6. Subscribe to the SQS queue.");
if (selectFIFO) {
    System.out.println(
        "If you add a filter to this subscription, then only the
filtered messages will be received in the queue.\n"
        +
        "For information about message filtering, see https://
docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n"
        +
        "For this example, you can filter messages by a \"tone\"
attribute.");
    System.out.println("Would you like to filter messages for " +
sqsQueueName + "'s subscription to the topic "
        + topicName + "? (y/n)");
    String filterAns = in.nextLine();
    if (filterAns.compareTo("y") == 0) {
        boolean moreAns = false;
        System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
        System.out.println("1. cheerful");
        System.out.println("2. funny");
        System.out.println("3. serious");
        System.out.println("4. sincere");
        while (!moreAns) {
            System.out.println("Select a number or choose 0 to end.");
            String ans = in.nextLine();
            switch (ans) {
                case "1":
                    filterList.add("cheerful");
                    break;
                case "2":
                    filterList.add("funny");
                    break;
                case "3":
                    filterList.add("serious");
                    break;
                case "4":
                    filterList.add("sincere");
                    break;
                default:
                    moreAns = true;
                    break;
            }
        }
    }
}
```



```
    }
  }
  subscriptionArn = subQueue(snsClient, topicArn, sqsQueueArn, filterList);
  System.out.println(DASHES);

  System.out.println(DASHES);
  System.out.println("7. Publish a message to the topic.");
  if (selectFIFO) {
    System.out.println("Would you like to add an attribute to this message?
(y/n)");
    String msgAns = in.nextLine();
    if (msgAns.compareTo("y") == 0) {
      System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
      System.out.println("1. cheerful");
      System.out.println("2. funny");
      System.out.println("3. serious");
      System.out.println("4. sincere");
      System.out.println("Select a number or choose 0 to end.");
      String ans = in.nextLine();
      switch (ans) {
        case "1":
          msgAttValue = "cheerful";
          break;
        case "2":
          msgAttValue = "funny";
          break;
        case "3":
          msgAttValue = "serious";
          break;
        default:
          msgAttValue = "sincere";
          break;
      }

      System.out.println("Selected value is " + msgAttValue);
    }
    System.out.println("Enter a message.");
    message = in.nextLine();
    pubMessageFIFO(snsClient, message, topicArn, msgAttValue, duplication,
groupId, deduplicationID);
  } else {
    System.out.println("Enter a message.");
```

```
        message = in.nextLine();
        pubMessage(snsClient, message, topicArn);
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("8. Display the message. Press any key to continue.");
    in.nextLine();
    messageList = receiveMessages(sqsClient, sqsQueueUrl, msgAttValue);
    for (Message mes : messageList) {
        System.out.println("Message Id: " + mes.messageId());
        System.out.println("Full Message: " + mes.body());
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("9. Delete the received message. Press any key to
continue.");
    in.nextLine();
    deleteMessages(sqsClient, sqsQueueUrl, messageList);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("10. Unsubscribe from the topic and delete the queue.
Press any key to continue.");
    in.nextLine();
    unSub(snsClient, subscriptionArn);
    deleteSQSQueue(sqsClient, sqsQueueName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("11. Delete the topic. Press any key to continue.");
    in.nextLine();
    deleteSNSTopic(snsClient, topicArn);

    System.out.println(DASHES);
    System.out.println("The SNS/SQS workflow has completed successfully.");
    System.out.println(DASHES);
}

public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
    try {
        DeleteTopicRequest request = DeleteTopicRequest.builder()
            .topicArn(topicArn)
```

```
        .build();

        DeleteTopicResponse result = snsClient.deleteTopic(request);
        System.out.println("Status was " +
result.sdkHttpResponse().statusCode());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
    try {
        GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
            .queueName(queueName)
            .build();

        String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
        DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
            .queueUrl(queueUrl)
            .build();

        sqsClient.deleteQueue(deleteQueueRequest);
        System.out.println(queueName + " was successfully deleted.");

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void unSub(SnsClient snsClient, String subscriptionArn) {
    try {
        UnsubscribeRequest request = UnsubscribeRequest.builder()
            .subscriptionArn(subscriptionArn)
            .build();

        UnsubscribeResponse result = snsClient.unsubscribe(request);
        System.out.println("Status was " + result.sdkHttpResponse().statusCode()
            + "\nSubscription was removed for " +
request.subscriptionArn());

    } catch (SnsException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
    try {
        List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();
        for (Message msg : messages) {
            DeleteMessageBatchRequestEntry entry =
DeleteMessageBatchRequestEntry.builder()
                .id(msg.messageId())
                .build();

            entries.add(entry);
        }

        DeleteMessageBatchRequest deleteMessageBatchRequest =
DeleteMessageBatchRequest.builder()
            .queueUrl(queueUrl)
            .entries(entries)
            .build();

        sqsClient.deleteMessageBatch(deleteMessageBatchRequest);
        System.out.println("The batch delete of messages was successful");

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl, String msgAttValue) {
    try {
        if (msgAttValue.isEmpty()) {
            ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                .queueUrl(queueUrl)
                .numberOfMessages(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 "";
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [CreateQueue](#)
 - [CreateTopic](#)
 - [DeleteMessageBatch](#)
 - [DeleteQueue](#)
 - [DeleteTopic](#)
 - [GetQueueAttributes](#)
 - [Publish](#)
 - [ReceiveMessage](#)
 - [SetQueueAttributes](#)

- [订阅](#)
- [Unsubscribe](#)

无服务器示例

从亚马逊触发器调用 Lambda 函数 SNS

以下代码示例说明如何实现一个 Lambda 函数，该函数接收通过接收来自主题的消息而触发的事件。SNS 该函数从事件参数检索消息并记录每条消息的内容。

SDK 适用于 Java 2.x

Note

还有更多相关信息 GitHub。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

使用 Java 在 Lambda 上使用 SNS 事件。

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

使用SDK适用于 Java 2.x 的亚马逊SQS示例

以下代码示例向您展示如何在 Amazon 中使用来执行操作和实现常见场景SQS。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

开始使用

你好 Amazon SQS

以下代码示例展示了如何开始使用 Amazon SQS。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

```
    }  
  }  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListQueues](#)”中的。

主题

- [操作](#)
- [场景](#)
- [无服务器示例](#)

操作

CreateQueue

以下代码示例演示如何使用 CreateQueue。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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());
        } catch (SqsException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```



```
        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)
```

```
        .numberOfMessages(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);
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateQueue](#)”中的。

DeleteMessage

以下代码示例演示如何使用 DeleteMessage。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteMessage](#)”中的。

DeleteQueue

以下代码示例演示如何使用 DeleteQueue。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteQueue](#)”中的。

GetQueueUrl

以下代码示例演示如何使用 `GetQueueUrl`。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
GetQueueUrlResponse getQueueUrlResponse = sqsClient
    .getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
    return getQueueUrlResponse.queueUrl();
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetQueueUrl](#)”中的。

ListQueues

以下代码示例演示如何使用 ListQueues。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListQueues](#)”中的。

ReceiveMessage

以下代码示例演示如何使用 ReceiveMessage。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
try {
    ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
        .queueUrl(queueUrl)
        .numberOfMessages(5)
        .build();
    return sqsClient.receiveMessage(receiveMessageRequest).messages();

} catch (SqsException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return null;
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ReceiveMessage](#)”中的。

SendMessage

以下代码示例演示如何使用 SendMessage。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。


```
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);
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [SendMessage](#)”中的。

SendMessageBatch

以下代码示例演示如何使用 SendMessageBatch。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [SendMessageBatch](#)”中的。

场景

创建并发布到FIFO主题

以下代码示例展示了如何创建和发布到 A FIFO mazon SNS 主题。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

此示例

- 创建一个 Amazon SNS FIFO 主题、两个亚马逊SQSFIFO队列和一个标准队列。
- 将队列订阅到主题，发布一条消息到主题。

该[测试](#)验证每个队列是否收到消息。[完整的示例](#)还显示了添加访问策略，并在最后删除了资源。

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {
```

```

    final String usage = "\n" +
        "Usage: " +
        "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
<analyticsQueueName>\n\n" +
        "Where:\n" +
        "    fifoTopicName - The name of the FIFO topic that you want to
create. \n\n" +
        "    wholesaleQueueARN - The name of a SQS FIFO queue that will be
created for the wholesale consumer. \n\n"
        +
        "    retailQueueARN - The name of a SQS FIFO queue that will created
for the retail consumer. \n\n" +
        "    analyticsQueueARN - The name of a SQS standard queue that will
be created for the analytics consumer. \n\n";
    if (args.length != 4) {
        System.out.println(usage);
        System.exit(1);
    }

    final String fifoTopicName = args[0];
    final String wholeSaleQueueName = args[1];
    final String retailQueueName = args[2];
    final String analyticsQueueName = args[3];

    // For convenience, the QueueData class holds metadata about a queue: ARN,
URL,
    // name and type.
    List<QueueData> queues = List.of(
        new QueueData(wholeSaleQueueName, QueueType.FIFO),
        new QueueData(retailQueueName, QueueType.FIFO),
        new QueueData(analyticsQueueName, QueueType.Standard));

    // Create queues.
    createQueues(queues);

    // Create a topic.
    String topicARN = createFIFOTopic(fifoTopicName);

    // Subscribe each queue to the topic.
    subscribeQueues(queues, topicARN);

    // Allow the newly created topic to send messages to the queues.
    addAccessPolicyToQueuesFINAL(queues, topicARN);

```

```
// Publish a sample price update message with payload.
publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

// Clean up resources.
deleteSubscriptions(queues);
deleteQueues(queues);
deleteTopic(topicARN);
}

public static String createFIFOtopic(String topicName) {
    try {
        // Create a FIFO topic by using the SNS service client.
        Map<String, String> topicAttributes = Map.of(
            "FifoTopic", "true",
            "ContentBasedDeduplication", "false");

        CreateTopicRequest topicRequest = CreateTopicRequest.builder()
            .name(topicName)
            .attributes(topicAttributes)
            .build();

        CreateTopicResponse response = snsClient.createTopic(topicRequest);
        String topicArn = response.topicArn();
        System.out.println("The topic ARN is" + topicArn);

        return topicArn;
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void subscribeQueues(List<QueueData> queues, String topicARN) {
    queues.forEach(queue -> {
        SubscribeRequest subscribeRequest = SubscribeRequest.builder()
            .topicArn(topicARN)
            .endpoint(queue.queueARN)
            .protocol("sqs")
            .build();
    });
}
```

```
        // Subscribe to the endpoint by using the SNS service client.
        // Only Amazon SQS queues can receive notifications from an Amazon SNS
FIFO
        // topic.
        SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
        System.out.println("The queue [" + queue.queueARN + "] subscribed to the
topic [" + topicARN + "]);
        queue.subscriptionARN = subscribeResponse.subscriptionArn();
    });
}

    public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

        try {
            // Create and publish a message that updates the wholesale price.
            String subject = "Price Update";
            String dedupId = UUID.randomUUID().toString();
            String attributeName = "business";
            String attributeValue = "wholesale";

            MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
                .dataType("String")
                .stringValue(attributeValue)
                .build();

            Map<String, MessageAttributeValue> attributes = new HashMap<>();
            attributes.put(attributeName, msgAttValue);
            PublishRequest pubRequest = PublishRequest.builder()
                .topicArn(topicArn)
                .subject(subject)
                .message(payload)
                .messageGroupId(groupId)
                .messageDeduplicationId(dedupId)
                .messageAttributes(attributes)
                .build();

            final PublishResponse response = snsClient.publish(pubRequest);
            System.out.println(response.messageId());
            System.out.println(response.sequenceNumber());
            System.out.println("Message was published to " + topicArn);

        } catch (SnsException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [CreateTopic](#)
 - [Publish](#)
 - [订阅](#)

处理 S3 事件通知

以下代码示例显示了如何以面向对象的方式处理 S3 事件通知。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

此示例说明如何使用 Amazon 处理 S3 通知事件SQS。

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

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [DeleteMessageBatch](#)
 - [GetQueueAttributes](#)
 - [PutBucketNotificationConfiguration](#)
 - [ReceiveMessage](#)

将消息发布到队列

以下代码示例展示了如何：

- 创建主题（FIFO或非-FIFO）。
- 针对主题订阅多个队列，并提供应用筛选条件的选项。
- 将消息发布到主题。
- 轮询队列中是否有收到的消息。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
package com.example.sns;
```

```
import
    software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.MessageAttributeValue;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import com.google.gson.Gson;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonPrimitive;

/**
 * Before running this Java V2 code example, set up your development
```

```
* environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*
* This Java example performs these tasks:
*
* 1. Gives the user three options to choose from.
* 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
* 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
* 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
* 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
* 6. Subscribes to the SQS queue.
* 7. Publishes a message to the topic.
* 8. Displays the messages.
* 9. Deletes the received message.
* 10. Unsubscribes from the topic.
* 11. Deletes the SNS topic.
*/
public class SNSWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:\n" +
            "    <fifoQueueARN>\n\n" +
            "Where:\n" +
            "    accountId - Your AWS account Id value.";

        // if (args.length != 1) {
        // System.out.println(usage);
        // System.exit(1);
        // }

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)

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

        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_EAST_1)
```

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

Scanner in = new Scanner(System.in);
String accountId = "814548047983";
String useFIFO;
String duplication = "n";
String topicName;
String deduplicationID = null;
String groupId = null;

String topicArn;
String sqsQueueName;
String sqsQueueUrl;
String sqsQueueArn;
String subscriptionArn;
boolean selectFIFO = false;

String message;
List<Message> messageList;
List<String> filterList = new ArrayList<>();
String msgAttValue = "";

System.out.println(DASHES);
System.out.println("Welcome to messaging with topics and queues.");
System.out.println("In this workflow, you will create an SNS topic and
subscribe an SQS queue to the topic.\n" +
    "You can select from several options for configuring the topic and
the subscriptions for the queue.\n" +
    "You can then post to the topic and see the results in the queue.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("SNS topics can be configured as FIFO (First-In-First-
Out).\n" +
    "FIFO topics deliver messages in order and support deduplication and
message filtering.\n" +
    "Would you like to work with FIFO topics? (y/n)");
useFIFO = in.nextLine();
if (useFIFO.compareTo("y") == 0) {
    selectFIFO = true;
    System.out.println("You have selected FIFO");
```

```
        System.out.println(" Because you have chosen a FIFO topic, deduplication
is supported.\n" +
        "          Deduplication IDs are either set in the message or
automatically generated from content using a hash function.\n"
        +
        "          If a message is successfully published to an SNS FIFO
topic, any message published and determined to have the same deduplication ID,\n"
        +
        "          within the five-minute deduplication interval, is
accepted but not delivered.\n" +
        "          For more information about deduplication, see https://
docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.");

        System.out.println(
            "Would you like to use content-based deduplication instead of
entering a deduplication ID? (y/n)");
        duplication = in.nextLine();
        if (duplication.compareTo("y") == 0) {
            System.out.println("Please enter a group id value");
            groupId = in.nextLine();
        } else {
            System.out.println("Please enter deduplication Id value");
            deduplicationID = in.nextLine();
            System.out.println("Please enter a group id value");
            groupId = in.nextLine();
        }
    }
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("2. Create a topic.");
System.out.println("Enter a name for your SNS topic.");
topicName = in.nextLine();
if (selectFIFO) {
    System.out.println("Because you have selected a FIFO topic, '.fifo' must
be appended to the topic name.");
    topicName = topicName + ".fifo";
    System.out.println("The name of the topic is " + topicName);
    topicArn = createFIFO(snsClient, topicName, duplication);
    System.out.println("The ARN of the FIFO topic is " + topicArn);

} else {
    System.out.println("The name of the topic is " + topicName);
    topicArn = createSNSTopic(snsClient, topicName);
}
```

```

        System.out.println("The ARN of the non-FIFO topic is " + topicArn);
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("3. Create an SQS queue.");
    System.out.println("Enter a name for your SQS queue.");
    sqsQueueName = in.nextLine();
    if (selectFIFO) {
        sqsQueueName = sqsQueueName + ".fifo";
    }
    sqsQueueUrl = createQueue(sqsClient, sqsQueueName, selectFIFO);
    System.out.println("The queue URL is " + sqsQueueUrl);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("4. Get the SQS queue ARN attribute.");
    sqsQueueArn = getSqsQueueAttrs(sqsClient, sqsQueueUrl);
    System.out.println("The ARN of the new queue is " + sqsQueueArn);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("5. Attach an IAM policy to the queue.");

    // Define the policy to use. Make sure that you change the REGION if you are
    // running this code
    // in a different region.
    String policy = "{\n" +
        "    \"Statement\": [\n" +
        "        {\n" +
        "            \"Effect\": \"Allow\", \n" +
        "            \"Principal\": {\n" +
        "                \"Service\": \"sns.amazonaws.com\"\n" +
        "            }, \n" +
        "            \"Action\": \"sqs:SendMessage\", \n" +
        "            \"Resource\": \"arn:aws:sqs:us-east-1:\" +
        accountId + ":" + sqsQueueName + "\", \n" +
        "            \"Condition\": {\n" +
        "                \"ArnEquals\": {\n" +
        "                    \"aws:SourceArn\": \"arn:aws:sns:us-east-1:\" +
        accountId + ":" + topicName + "\"\n" +
        "                }\n" +
        "            }\n" +
        "        }\n" +
    ";

```

```
        "    }\n" +  
        "    ]\n" +  
        " }";  
  
    setQueueAttr(sqsClient, sqsQueueUrl, policy);  
    System.out.println(DASHES);  
  
    System.out.println(DASHES);  
    System.out.println("6. Subscribe to the SQS queue.");  
    if (selectFIFO) {  
        System.out.println(  
            "If you add a filter to this subscription, then only the  
filtered messages will be received in the queue.\n"  
            +  
            "For information about message filtering, see https://  
docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n"  
            +  
            "For this example, you can filter messages by a \"tone\"  
attribute.");  
        System.out.println("Would you like to filter messages for " +  
sqsQueueName + "'s subscription to the topic "  
            + topicName + "? (y/n)");  
        String filterAns = in.nextLine();  
        if (filterAns.compareTo("y") == 0) {  
            boolean moreAns = false;  
            System.out.println("You can filter messages by one or more of the  
following \"tone\" attributes.");  
            System.out.println("1. cheerful");  
            System.out.println("2. funny");  
            System.out.println("3. serious");  
            System.out.println("4. sincere");  
            while (!moreAns) {  
                System.out.println("Select a number or choose 0 to end.");  
                String ans = in.nextLine();  
                switch (ans) {  
                    case "1":  
                        filterList.add("cheerful");  
                        break;  
                    case "2":  
                        filterList.add("funny");  
                        break;  
                    case "3":  
                        filterList.add("serious");  
                        break;
```



```
        case "4":
            filterList.add("sincere");
            break;
        default:
            moreAns = true;
            break;
    }
}
}
}
subscriptionArn = subQueue(snsClient, topicArn, sqsQueueArn, filterList);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Publish a message to the topic.");
if (selectFIFO) {
    System.out.println("Would you like to add an attribute to this message?
(y/n)");
    String msgAns = in.nextLine();
    if (msgAns.compareTo("y") == 0) {
        System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
        System.out.println("1. cheerful");
        System.out.println("2. funny");
        System.out.println("3. serious");
        System.out.println("4. sincere");
        System.out.println("Select a number or choose 0 to end.");
        String ans = in.nextLine();
        switch (ans) {
            case "1":
                msgAttValue = "cheerful";
                break;
            case "2":
                msgAttValue = "funny";
                break;
            case "3":
                msgAttValue = "serious";
                break;
            default:
                msgAttValue = "sincere";
                break;
        }

        System.out.println("Selected value is " + msgAttValue);
    }
}
```

```
    }
    System.out.println("Enter a message.");
    message = in.nextLine();
    pubMessageFIFO(snsClient, message, topicArn, msgAttValue, duplication,
groupId, deduplicationID);

} else {
    System.out.println("Enter a message.");
    message = in.nextLine();
    pubMessage(snsClient, message, topicArn);
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Display the message. Press any key to continue.");
in.nextLine();
messageList = receiveMessages(sqsClient, sqsQueueUrl, msgAttValue);
for (Message mes : messageList) {
    System.out.println("Message Id: " + mes.messageId());
    System.out.println("Full Message: " + mes.body());
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Delete the received message. Press any key to
continue.");
in.nextLine();
deleteMessages(sqsClient, sqsQueueUrl, messageList);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Unsubscribe from the topic and delete the queue.
Press any key to continue.");
in.nextLine();
unSub(snsClient, subscriptionArn);
deleteSQSQueue(sqsClient, sqsQueueName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Delete the topic. Press any key to continue.");
in.nextLine();
deleteSNSTopic(snsClient, topicArn);

System.out.println(DASHES);
```

```
        System.out.println("The SNS/SQS workflow has completed successfully.");
        System.out.println(DASHES);
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                .topicArn(topicArn)
                .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("Status was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                .queueName(queueName)
                .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                .queueUrl(queueUrl)
                .build();

            sqsClient.deleteQueue(deleteQueueRequest);
            System.out.println(queueName + " was successfully deleted.");

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                .subscriptionArn(subscriptionArn)
```

```
        .build();

        UnsubscribeResponse result = snsClient.unsubscribe(request);
        System.out.println("Status was " + result.sdkHttpResponse().statusCode()
            + "\nSubscription was removed for " +
request.subscriptionArn());

    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
    try {
        List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();
        for (Message msg : messages) {
            DeleteMessageBatchRequestEntry entry =
DeleteMessageBatchRequestEntry.builder()
                .id(msg.messageId())
                .build();

            entries.add(entry);
        }

        DeleteMessageBatchRequest deleteMessageBatchRequest =
DeleteMessageBatchRequest.builder()
            .queueUrl(queueUrl)
            .entries(entries)
            .build();

        sqsClient.deleteMessageBatch(deleteMessageBatchRequest);
        System.out.println("The batch delete of messages was successful");

    } catch (SqsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl, String msgAttValue) {
    try {
```

```
        if (msgAttValue.isEmpty()) {
            ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                .queueUrl(queueUrl)
                .numberOfMessages(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.
                .numberOfMessages(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 "";
}
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。

- [CreateQueue](#)
- [CreateTopic](#)
- [DeleteMessageBatch](#)
- [DeleteQueue](#)

- [DeleteTopic](#)
- [GetQueueAttributes](#)
- [Publish](#)
- [ReceiveMessage](#)
- [SetQueueAttributes](#)
- [订阅](#)
- [Unsubscribe](#)

无服务器示例

从亚马逊触发器调用 Lambda 函数 SQS

以下代码示例说明如何实现一个 Lambda 函数，该函数接收通过从队列接收消息而触发的事件。SQS 该函数从事件参数检索消息并记录每条消息的内容。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

使用 Java 在 Lambda 上使用SQS事件。

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

使用亚马逊触发器报告 Lambda 函数的批量项目失败 SQS

以下代码示例说明如何为从队列接收事件的 Lambda 函数实现部分批量响应。SQS 该函数在响应中报告批处理项目失败，并指示 Lambda 稍后重试这些消息。

SDK适用于 Java 2.x

Note

还有更多相关信息 [GitHub](#)。在[无服务器示例](#)存储库中查找完整示例，并了解如何进行设置和运行。

使用 Java 使用 Lambda 报告 SQS 批处理项目失败。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.amazonaws.services.lambda.runtime.events.SQSBatchResponse;

import java.util.ArrayList;
```

```
import java.util.List;

public class ProcessSQSMessageBatch implements RequestHandler<SQSEvent,
SQSBatchResponse> {
    @Override
    public SQSBatchResponse handleRequest(SQSEvent sqsEvent, Context context) {

        List<SQSBatchResponse.BatchItemFailure> batchItemFailures = new
        ArrayList<SQSBatchResponse.BatchItemFailure>();
        String messageId = "";
        for (SQSEvent.SQSMessage message : sqsEvent.getRecords()) {
            try {
                //process your message
                messageId = message.getMessageId();
            } catch (Exception e) {
                //Add failed message identifier to the batchItemFailures list
                batchItemFailures.add(new
                SQSBatchResponse.BatchItemFailure(messageId));
            }
        }
        return new SQSBatchResponse(batchItemFailures);
    }
}
```

用SDK于 Java 2.x 的 Step Functions 示例

以下代码示例向您展示了如何使用 with Step Functions 来执行操作和实现常见场景。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

开始使用

Hello Step Functions

以下代码示例展示了如何开始使用 Step Functions。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

Java 版本的 Hello。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
import software.amazon.awssdk.services.sfn.model.ListStateMachinesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.StateMachineListItem;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListStateMachines {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
            .region(region)
            .build();

        listMachines(sfnClient);
        sfnClient.close();
    }

    public static void listMachines(SfnClient sfnClient) {
        try {
            ListStateMachinesResponse response = sfnClient.listStateMachines();
            List<StateMachineListItem> machines = response.stateMachines();
            for (StateMachineListItem machine : machines) {
                System.out.println("The name of the state machine is: " +
                    machine.name());
            }
        }
    }
}
```

```
        System.out.println("The ARN value is : " +
machine.stateMachineArn());
    }

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListStateMachines](#)”中的。

主题

- [操作](#)
- [场景](#)

操作

CreateActivity

以下代码示例演示如何使用 CreateActivity。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateActivity](#)”中的。

CreateStateMachine

以下代码示例演示如何使用 CreateStateMachine。

SDK适用于 Java 2.x

Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateStateMachine](#)”中的。

DeleteActivity

以下代码示例演示如何使用 DeleteActivity。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。


```
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);  
    }  
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteActivity](#)”中的。

DeleteStateMachine

以下代码示例演示如何使用 DeleteStateMachine。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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.");
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteStateMachine](#)”中的。

DescribeExecution

以下代码示例演示如何使用 DescribeExecution。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeExecution](#)”中的。

DescribeStateMachine

以下代码示例演示如何使用 DescribeStateMachine。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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());
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeStateMachine](#)”中的。

GetActivityTask

以下代码示例演示如何使用 `GetActivityTask`。

SDK适用于 Java 2.x

Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public static List<String> getActivityTask(SfnClient sfnClient, String actArn) {
    List<String> myList = new ArrayList<>();
    GetActivityTaskRequest getActivityTaskRequest =
    GetActivityTaskRequest.builder()
        .activityArn(actArn)
        .build();

    GetActivityTaskResponse response =
    sfnClient.getActivityTask(getActivityTaskRequest);
    myList.add(response.taskToken());
    myList.add(response.input());
    return myList;
}

/// <summary>
/// Stop execution of a Step Functions workflow.
/// </summary>
/// <param name="executionArn">The Amazon Resource Name (ARN) of
/// the Step Functions execution to stop.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> StopExecution(string executionArn)
{
    var response =
        await _amazonStepFunctions.StopExecutionAsync(new StopExecutionRequest
    { ExecutionArn = executionArn });
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [GetActivityTask](#)”中的。

ListActivities

以下代码示例演示如何使用 ListActivities。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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();

        listAllActivities(sfnClient);
        sfnClient.close();
    }

    public static void listAllActivities(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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListActivities](#)”中的。

ListExecutions

以下代码示例演示如何使用 ListExecutions。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListExecutions](#)”中的。

ListStateMachines

以下代码示例演示如何使用 ListStateMachines。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListStateMachines](#)”中的。

SendTaskSuccess

以下代码示例演示如何使用 SendTaskSuccess。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [SendTaskSuccess](#)”中的。

StartExecution

以下代码示例演示如何使用 StartExecution。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartExecution](#)”中的。

场景

开始使用状态机

以下代码示例演示了操作流程：

- 创建活动。
- 根据包含先前创建的活动作为步骤的 Amazon States Language 定义创建状态机。
- 运行状态机并使用用户输入响应活动。
- 运行完成后获取最终状态和输出，然后清理资源。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**
 * You can obtain the JSON file to create a state machine in the following
 * GitHub location.
 *
 * https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/resources/sample_files
 *
 * To run this code example, place the chat_sfn_state_machine.json file into
 * your project's resources folder.
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code example performs the following tasks:
 *
 * 1. Creates an activity.
 * 2. Creates a state machine.
 * 3. Describes the state machine.
 * 4. Starts execution of the state machine and interacts with it.
 * 5. Describes the execution.
 * 6. Delete the activity.
 * 7. Deletes the state machine.
 */
public class StepFunctionsScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws Exception {
        final String usage = ""

            Usage:
                <roleARN> <activityName> <stateMachineName>
```

```

        Where:
            roleName - The name of the IAM role to create for this state
machine.
            activityName - The name of an activity to create.
            stateMachineName - The name of the state machine to create.
        """;

    if (args.length != 3) {
        System.out.println(usage);
        System.exit(1);
    }

    String roleName = args[0];
    String activityName = args[1];
    String stateMachineName = args[2];
    String polJSON = "{\n" +
        "    \"Version\": \"2012-10-17\",\n" +
        "    \"Statement\": [\n" +
        "        {\n" +
        "            \"Sid\": \"\",\n" +
        "            \"Effect\": \"Allow\",\n" +
        "            \"Principal\": {\n" +
        "                \"Service\": \"states.amazonaws.com\"\n" +
        "            },\n" +
        "            \"Action\": \"sts:AssumeRole\"\n" +
        "        }\n" +
        "    ]\n" +
        "}";

    Scanner sc = new Scanner(System.in);
    boolean action = false;

    Region region = Region.US_EAST_1;
    SfnClient sfnClient = SfnClient.builder()
        .region(region)
        .build();

    Region regionGl = Region.AWS_GLOBAL;
    IamClient iam = IamClient.builder()
        .region(regionGl)
        .build();

    System.out.println(DASHES);
    System.out.println("Welcome to the AWS Step Functions example scenario.");

```

```
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create an activity.");
String activityArn = createActivity(sfnClient, activityName);
System.out.println("The ARN of the activity is " + activityArn);
System.out.println(DASHES);

// Get JSON to use for the state machine and place the activityArn value
into
// it.
InputStream input = StepFunctionsScenario.class.getClassLoader()
    .getResourceAsStream("chat_sfn_state_machine.json");
ObjectMapper mapper = new ObjectMapper();
JsonNode jsonNode = mapper.readValue(input, JsonNode.class);
String jsonString = mapper.writeValueAsString(jsonNode);

// Modify the Resource node.
ObjectMapper objectMapper = new ObjectMapper();
JsonNode root = objectMapper.readTree(jsonString);
((ObjectNode) root.path("States").path("GetInput")).put("Resource",
activityArn);

// Convert the modified Java object back to a JSON string.
String stateDefinition = objectMapper.writeValueAsString(root);
System.out.println(stateDefinition);

System.out.println(DASHES);
System.out.println("2. Create a state machine.");
String roleARN = createIAMRole(iam, roleName, polJSON);
String stateMachineArn = createMachine(sfnClient, roleARN, stateMachineName,
stateDefinition);
System.out.println("The ARN of the state machine is " + stateMachineArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. Describe the state machine.");
describeStateMachine(sfnClient, stateMachineArn);
System.out.println("What should ChatSFN call you?");
String userName = sc.nextLine();
System.out.println("Hello " + userName);
System.out.println(DASHES);

System.out.println(DASHES);
```

```
// The JSON to pass to the StartExecution call.
String executionJson = "{ \"name\" : \"" + userName + "\" }";
System.out.println(executionJson);
System.out.println("4. Start execution of the state machine and interact
with it.");
String runArn = startWorkflow(sfnClient, stateMachineArn, executionJson);
System.out.println("The ARN of the state machine execution is " + runArn);
List<String> myList;
while (!action) {
    myList = getActivityTask(sfnClient, activityArn);
    System.out.println("ChatSFN: " + myList.get(1));
    System.out.println(userName + " please specify a value.");
    String myAction = sc.nextLine();
    if (myAction.compareTo("done") == 0)
        action = true;

    System.out.println("You have selected " + myAction);
    String taskJson = "{ \"action\" : \"" + myAction + "\" }";
    System.out.println(taskJson);
    sendTaskSuccess(sfnClient, myList.get(0), taskJson);
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Describe the execution.");
describeExe(sfnClient, runArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Delete the activity.");
deleteActivity(sfnClient, activityArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Delete the state machines.");
deleteMachine(sfnClient, stateMachineArn);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("The AWS Step Functions example scenario is complete.");
System.out.println(DASHES);
}
```



```
public static String createIAMRole(IamClient iam, String rolename, String
polJSON) {
    try {
        CreateRoleRequest request = CreateRoleRequest.builder()
            .roleName(rolename)
            .assumeRolePolicyDocument(polJSON)
            .description("Created using the AWS SDK for Java")
            .build();

        CreateRoleResponse response = iam.createRole(request);
        return response.role().arn();

    } catch (IamException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static void describeExe(SfnClient sfnClient, String executionArn) {
    try {
        DescribeExecutionRequest executionRequest =
DescribeExecutionRequest.builder()
            .executionArn(executionArn)
            .build();

        String status = "";
        boolean hasSucceeded = false;
        while (!hasSucceeded) {
            DescribeExecutionResponse response =
sfnClient.describeExecution(executionRequest);
            status = response.statusAsString();
            if (status.compareTo("RUNNING") == 0) {
                System.out.println("The state machine is still running, let's
wait for it to finish.");
                Thread.sleep(2000);
            } else if (status.compareTo("SUCCEEDED") == 0) {
                System.out.println("The Step Function workflow has succeeded");
                hasSucceeded = true;
            } else {
                System.out.println("The Status is neither running or
succeeded");
            }
        }
    }
}
```

```
        System.out.println("The Status is " + status);

    } catch (SfnException | InterruptedException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public static void sendTaskSuccess(SfnClient sfnClient, String token, String
json) {
    try {
        SendTaskSuccessRequest successRequest = SendTaskSuccessRequest.builder()
            .taskToken(token)
            .output(json)
            .build();

        sfnClient.sendTaskSuccess(successRequest);

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static List<String> getActivityTask(SfnClient sfnClient, String actArn) {
    List<String> myList = new ArrayList<>();
    GetActivityTaskRequest getActivityTaskRequest =
GetActivityTaskRequest.builder()
        .activityArn(actArn)
        .build();

    GetActivityTaskResponse response =
sfnClient.getActivityTask(getActivityTaskRequest);
    myList.add(response.taskToken());
    myList.add(response.input());
    return myList;
}

public static void deleteActivity(SfnClient sfnClient, String actArn) {
    try {
        DeleteActivityRequest activityRequest = DeleteActivityRequest.builder()
            .activityArn(actArn)
            .build();
```

```
        sfnClient.deleteActivity(activityRequest);
        System.out.println("You have deleted " + actArn);

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void describeStateMachine(SfnClient sfnClient, String
stateMachineArn) {
    try {
        DescribeStateMachineRequest stateMachineRequest =
DescribeStateMachineRequest.builder()
            .stateMachineArn(stateMachineArn)
            .build();

        DescribeStateMachineResponse response =
sfnClient.describeStateMachine(stateMachineRequest);
        System.out.println("The name of the State machine is " +
response.name());
        System.out.println("The status of the State machine is " +
response.status());
        System.out.println("The ARN value of the State machine is " +
response.stateMachineArn());
        System.out.println("The role ARN value is " + response.roleArn());

    } catch (SfnException e) {
        System.err.println(e.getMessage());
    }
}

public static void deleteMachine(SfnClient sfnClient, String stateMachineArn) {
    try {
        DeleteStateMachineRequest deleteStateMachineRequest =
DeleteStateMachineRequest.builder()
            .stateMachineArn(stateMachineArn)
            .build();

        sfnClient.deleteStateMachine(deleteStateMachineRequest);
        DescribeStateMachineRequest describeStateMachine =
DescribeStateMachineRequest.builder()
            .stateMachineArn(stateMachineArn)
            .build();
```

```
        while (true) {
            DescribeStateMachineResponse response =
sfnClient.describeStateMachine(describeStateMachine);
            System.out.println("The state machine is not deleted yet. The status
is " + response.status());
            Thread.sleep(3000);
        }

    } catch (SfnException | InterruptedException e) {
        System.err.println(e.getMessage());
    }
    System.out.println(stateMachineArn + " was successfully deleted.");
}

public static String startWorkflow(SfnClient sfnClient, String stateMachineArn,
String jsonEx) {
    UUID uuid = UUID.randomUUID();
    String uuidValue = uuid.toString();
    try {
        StartExecutionRequest executionRequest = StartExecutionRequest.builder()
            .input(jsonEx)
            .stateMachineArn(stateMachineArn)
            .name(uuidValue)
            .build();

        StartExecutionResponse response =
sfnClient.startExecution(executionRequest);
        return response.executionArn();

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static String createMachine(SfnClient sfnClient, String roleARN, String
stateMachineName, String json) {
    try {
        CreateStateMachineRequest machineRequest =
CreateStateMachineRequest.builder()
            .definition(json)
            .name(stateMachineName)
```

```
        .roleArn(roleARN)
        .type(StateMachineType.STANDARD)
        .build();

    CreateStateMachineResponse response =
sfnClient.createStateMachine(machineRequest);
    return response.stateMachineArn();

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

public static String createActivity(SfnClient sfnClient, String activityName) {
    try {
        CreateActivityRequest activityRequest = CreateActivityRequest.builder()
            .name(activityName)
            .build();

        CreateActivityResponse response =
sfnClient.createActivity(activityRequest);
        return response.activityArn();

    } catch (SfnException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [CreateActivity](#)
 - [CreateStateMachine](#)
 - [DeleteActivity](#)
 - [DeleteStateMachine](#)
 - [DescribeExecution](#)
 - [DescribeStateMachine](#)

- [GetActivityTask](#)
- [ListActivities](#)
- [ListStateMachines](#)
- [SendTaskSuccess](#)
- [StartExecution](#)
- [StopExecution](#)

AWS STS 用SDK于 Java 2.x 的示例

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS STS。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

AssumeRole

以下代码示例演示如何使用 AssumeRole。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

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

```
import software.amazon.awssdk.services.sts.model.AssumeRoleRequest;
import software.amazon.awssdk.services.sts.model.StsException;
import software.amazon.awssdk.services.sts.model.AssumeRoleResponse;
import software.amazon.awssdk.services.sts.model.Credentials;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * To make this code example work, create a Role that you want to assume.
 * Then define a Trust Relationship in the AWS Console. You can use this as an
 * example:
 *
 * {
 *   "Version": "2012-10-17",
 *   "Statement": [
 *     {
 *       "Effect": "Allow",
 *       "Principal": {
 *         "AWS": "<Specify the ARN of your IAM user you are using in this code
 * example>"
 *       },
 *       "Action": "sts:AssumeRole"
 *     }
 *   ]
 * }
 *
 * For more information, see "Editing the Trust Relationship for an Existing
 * Role" in the AWS Directory Service guide.
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AssumeRole {
    public static void main(String[] args) {
        final String usage = ""

                Usage:
                <roleArn> <roleSessionName>\s
```

```
        Where:
            roleArn - The Amazon Resource Name (ARN) of the role to assume
(for example, rn:aws:iam::000008047983:role/s3role).\s
            roleSessionName - An identifier for the assumed role session
(for example, mysession).\s
            """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String roleArn = args[0];
    String roleSessionName = args[1];
    Region region = Region.US_EAST_1;
    StsClient stsClient = StsClient.builder()
        .region(region)
        .build();

    assumeGivenRole(stsClient, roleArn, roleSessionName);
    stsClient.close();
}

public static void assumeGivenRole(StsClient stsClient, String roleArn, String
roleSessionName) {
    try {
        AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
            .roleArn(roleArn)
            .roleSessionName(roleSessionName)
            .build();

        AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
        Credentials myCreds = roleResponse.credentials();

        // Display the time when the temp creds expire.
        Instant exTime = myCreds.expiration();
        String tokenInfo = myCreds.sessionToken();

        // Convert the Instant to readable date.
        DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
            .withLocale(Locale.US)
            .withZone(ZoneId.systemDefault());
```



```
        formatter.format(exTime);
        System.out.println("The token " + tokenInfo + " expires on " + exTime);

    } catch (StsException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [AssumeRole](#)”中的。

AWS Support 用SDK于 Java 2.x 的示例

以下代码示例向您展示了如何使用with来执行操作和实现常见场景 AWS Support。AWS SDK for Java 2.x

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

开始使用

你好 AWS Support

以下代码示例展示了如何开始使用 AWS Support。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeServices](#)”中的。

主题

- [操作](#)
- [场景](#)

操作

AddAttachmentsToSet

以下代码示例演示如何使用 AddAttachmentsToSet。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [AddAttachmentsToSet](#)”中的。

AddCommunicationToCase

以下代码示例演示如何使用 AddCommunicationToCase。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [AddCommunicationToCase](#)”中的。

CreateCase

以下代码示例演示如何使用 CreateCase。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateCase](#)”中的。

DescribeAttachment

以下代码示例演示如何使用 DescribeAttachment。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeAttachment](#)”中的。

DescribeCases

以下代码示例演示如何使用 DescribeCases。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeCases](#)”中的。

DescribeCommunications

以下代码示例演示如何使用 DescribeCommunications。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeCommunications](#)”中的。

DescribeServices

以下代码示例演示如何使用 DescribeServices。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
// 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;
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeServices](#)”中的。

DescribeSeverityLevels

以下代码示例演示如何使用 DescribeSeverityLevels。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 "";
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeSeverityLevels](#)”中的。

ResolveCase

以下代码示例演示如何使用 ResolveCase。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ResolveCase](#)”中的。

场景

开始应用场景

以下代码示例演示了操作流程：

- 获取并显示案例的可用服务和严重级别。
- 使用选定的服务、类别和严重性级别创建支持案例。
- 获取并显示当天打开案例的列表。
- 向新案例添加附件集和通信。
- 描述该案例的新附件和通信。
- 解析案例。
- 获取并显示当天未解决的案例列表。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

运行各种 AWS Support 操作。

```
import software.amazon.awssdk.core.SdkBytes;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.support.SupportClient;
import software.amazon.awssdk.services.support.model.AddAttachmentsToSetResponse;
import software.amazon.awssdk.services.support.model.AddCommunicationToCaseRequest;
import software.amazon.awssdk.services.support.model.AddCommunicationToCaseResponse;
import software.amazon.awssdk.services.support.model.Attachment;
import software.amazon.awssdk.services.support.model.AttachmentDetails;
import software.amazon.awssdk.services.support.model.CaseDetails;
import software.amazon.awssdk.services.support.model.Category;
import software.amazon.awssdk.services.support.model.Communication;
import software.amazon.awssdk.services.support.model.CreateCaseRequest;
import software.amazon.awssdk.services.support.model.CreateCaseResponse;
import software.amazon.awssdk.services.support.model.DescribeAttachmentRequest;
import software.amazon.awssdk.services.support.model.DescribeAttachmentResponse;
import software.amazon.awssdk.services.support.model.DescribeCasesRequest;
import software.amazon.awssdk.services.support.model.DescribeCasesResponse;
import software.amazon.awssdk.services.support.model.DescribeCommunicationsRequest;
import software.amazon.awssdk.services.support.model.DescribeCommunicationsResponse;
import software.amazon.awssdk.services.support.model.DescribeServicesRequest;
import software.amazon.awssdk.services.support.model.DescribeServicesResponse;
import software.amazon.awssdk.services.support.model.DescribeSeverityLevelsRequest;
import software.amazon.awssdk.services.support.model.DescribeSeverityLevelsResponse;
import software.amazon.awssdk.services.support.model.ResolveCaseRequest;
import software.amazon.awssdk.services.support.model.ResolveCaseResponse;
import software.amazon.awssdk.services.support.model.Service;
import software.amazon.awssdk.services.support.model.SeverityLevel;
import software.amazon.awssdk.services.support.model.SupportException;
import software.amazon.awssdk.services.support.model.AddAttachmentsToSetRequest;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.time.Instant;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html

```

```
*
* In addition, you must have the AWS Business Support Plan to use the AWS
* Support Java API. For more information, see:
*
* https://aws.amazon.com/premiumsupport/plans/
*
* This Java example performs the following tasks:
*
* 1. Gets and displays available services.
* 2. Gets and displays severity levels.
* 3. Creates a support case by using the selected service, category, and
* severity level.
* 4. Gets a list of open cases for the current day.
* 5. Creates an attachment set with a generated file.
* 6. Adds a communication with the attachment to the support case.
* 7. Lists the communications of the support case.
* 8. Describes the attachment set included with the communication.
* 9. Resolves the support case.
* 10. Gets a list of resolved cases for the current day.
*/
public class SupportScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = ""

            Usage:
            <fileAttachment>Where:
            fileAttachment - The file can be a simple saved .txt file to use
as an email attachment.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String fileAttachment = args[0];
        Region region = Region.US_WEST_2;
        SupportClient supportClient = SupportClient.builder()
            .region(region)
            .build();
    }
}
```

```
        System.out.println(DASHES);
        System.out.println("***** Welcome to the AWS Support case example
scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Get and display available services.");
        List<String> sevCatList = displayServices(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Get and display Support severity levels.");
        String sevLevel = displaySevLevels(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create a support case using the selected service,
category, and severity level.");
        String caseId = createSupportCase(supportClient, sevCatList, sevLevel);
        if (caseId.compareTo("") == 0) {
            System.out.println("A support case was not successfully created!");
            System.exit(1);
        } else
            System.out.println("Support case " + caseId + " was successfully
created!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get open support cases.");
        getOpenCase(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Create an attachment set with a generated file to add
to the case.");
        String attachmentSetId = addAttachment(supportClient, fileAttachment);
        System.out.println("The Attachment Set id value is" + attachmentSetId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Add communication with the attachment to the support
case.");
        addAttachSupportCase(supportClient, caseId, attachmentSetId);
        System.out.println(DASHES);
```



```
        System.out.println(DASHES);
        System.out.println("7. List the communications of the support case.");
        String attachId = listCommunications(supportClient, caseId);
        System.out.println("The Attachment id value is" + attachId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Describe the attachment set included with the
communication.");
        describeAttachment(supportClient, attachId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Resolve the support case.");
        resolveSupportCase(supportClient, caseId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get a list of resolved cases for the current day.");
        getResolvedCase(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("***** This Scenario has successfully completed");
        System.out.println(DASHES);
    }

    public static void getResolvedCase(SupportClient supportClient) {
        try {
            // Specify the start and end time.
            Instant now = Instant.now();
            java.time.LocalDate.now();
            Instant yesterday = now.minus(1, ChronoUnit.DAYS);

            DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
                .maxResults(30)
                .afterTime(yesterday.toString())
                .beforeTime(now.toString())
                .includeResolvedCases(true)
                .build();
```

```
        DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
        List<CaseDetails> cases = response.cases();
        for (CaseDetails sinCase : cases) {
            if (sinCase.status().compareTo("resolved") == 0)
                System.out.println("The case status is " + sinCase.status());
        }

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void resolveSupportCase(SupportClient supportClient, String
caseId) {
    try {
        ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()
            .caseId(caseId)
            .build();

        ResolveCaseResponse response = supportClient.resolveCase(caseRequest);
        System.out.println("The status of case " + caseId + " is " +
response.finalCaseStatus());

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static void describeAttachment(SupportClient supportClient, String
attachId) {
    try {
        DescribeAttachmentRequest attachmentRequest =
DescribeAttachmentRequest.builder()
            .attachmentId(attachId)
            .build();

        DescribeAttachmentResponse response =
supportClient.describeAttachment(attachmentRequest);
        System.out.println("The name of the file is " +
response.attachment().fileName());
    }
}
```

```
        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static String listCommunications(SupportClient supportClient, String
caseId) {
        try {
            String attachId = null;
            DescribeCommunicationsRequest communicationsRequest =
DescribeCommunicationsRequest.builder()
                .caseId(caseId)
                .maxResults(10)
                .build();

            DescribeCommunicationsResponse response =
supportClient.describeCommunications(communicationsRequest);
            List<Communication> communications = response.communications();
            for (Communication comm : communications) {
                System.out.println("the body is: " + comm.body());

                // Get the attachment id value.
                List<AttachmentDetails> attachments = comm.attachmentSet();
                for (AttachmentDetails detail : attachments) {
                    attachId = detail.attachmentId();
                }
            }
            return attachId;
        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    public static void addAttachSupportCase(SupportClient supportClient, String
caseId, String attachmentSetId) {
        try {
            AddCommunicationToCaseRequest caseRequest =
AddCommunicationToCaseRequest.builder()
                .caseId(caseId)
                .attachmentSetId(attachmentSetId)
```

```
        .communicationBody("Please refer to attachment for details.")
        .build();

        AddCommunicationToCaseResponse response =
supportClient.addCommunicationToCase(caseRequest);
        if (response.result())
            System.out.println("You have successfully added a communication to
an AWS Support case");
        else
            System.out.println("There was an error adding the communication to
an AWS Support case");

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static String addAttachment(SupportClient supportClient, String
fileAttachment) {
    try {
        File myFile = new File(fileAttachment);
        InputStream sourceStream = new FileInputStream(myFile);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

        Attachment attachment = Attachment.builder()
            .fileName(myFile.getName())
            .data(sourceBytes)
            .build();

        AddAttachmentsToSetRequest setRequest =
AddAttachmentsToSetRequest.builder()
            .attachments(attachment)
            .build();

        AddAttachmentsToSetResponse response =
supportClient.addAttachmentsToSet(setRequest);
        return response.attachmentSetId();

    } catch (SupportException | FileNotFoundException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}
```

```
}

public static void getOpenCase(SupportClient supportClient) {
    try {
        // Specify the start and end time.
        Instant now = Instant.now();
        java.time.LocalDate.now();
        Instant yesterday = now.minus(1, ChronoUnit.DAYS);

        DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
            .maxResults(20)
            .afterTime(yesterday.toString())
            .beforeTime(now.toString())
            .build();

        DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
        List<CaseDetails> cases = response.cases();
        for (CaseDetails sinCase : cases) {
            System.out.println("The case status is " + sinCase.status());
            System.out.println("The case Id is " + sinCase.caseId());
            System.out.println("The case subject is " + sinCase.subject());
        }

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
}

public static String createSupportCase(SupportClient supportClient, List<String>
sevCatList, String sevLevel) {
    try {
        String serviceCode = sevCatList.get(0);
        String caseCat = sevCatList.get(1);
        CreateCaseRequest caseRequest = CreateCaseRequest.builder()
            .categoryCode(caseCat.toLowerCase())
            .serviceCode(serviceCode.toLowerCase())
            .severityCode(sevLevel.toLowerCase())
            .communicationBody("Test issue with " +
serviceCode.toLowerCase())
            .subject("Test case, please ignore")
            .language("en")
    }
}
```

```
        .issueType("technical")
        .build();

        CreateCaseResponse response = supportClient.createCase(caseRequest);
        return response.caseId();

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

public static String displaySevLevels(SupportClient supportClient) {
    try {
        DescribeSeverityLevelsRequest severityLevelsRequest =
DescribeSeverityLevelsRequest.builder()
            .language("en")
            .build();

        DescribeSeverityLevelsResponse response =
supportClient.describeSeverityLevels(severityLevelsRequest);
        List<SeverityLevel> severityLevels = response.severityLevels();
        String levelName = null;
        for (SeverityLevel sevLevel : severityLevels) {
            System.out.println("The severity level name is: " +
sevLevel.name());
            if (sevLevel.name().compareTo("High") == 0)
                levelName = sevLevel.name();
        }
        return levelName;

    } catch (SupportException e) {
        System.out.println(e.getLocalizedMessage());
        System.exit(1);
    }
    return "";
}

// Return a List that contains a Service name and Category name.
public static List<String> displayServices(SupportClient supportClient) {
    try {
        DescribeServicesRequest servicesRequest =
DescribeServicesRequest.builder()
```

```
        .language("en")
        .build();

    DescribeServicesResponse response =
supportClient.describeServices(servicesRequest);
    String serviceCode = null;
    String catName = null;
    List<String> sevCatList = new ArrayList<>();
    List<Service> services = response.services();

    System.out.println("Get the first 10 services");
    int index = 1;
    for (Service service : services) {
        if (index == 11)
            break;

        System.out.println("The Service name is: " + service.name());
        if (service.name().compareTo("Account") == 0)
            serviceCode = service.code();

        // Get the Categories for this service.
        List<Category> categories = service.categories();
        for (Category cat : categories) {
            System.out.println("The category name is: " + cat.name());
            if (cat.name().compareTo("Security") == 0)
                catName = cat.name();
        }
        index++;
    }

    // Push the two values to the list.
    sevCatList.add(serviceCode);
    sevCatList.add(catName);
    return sevCatList;

} catch (SupportException e) {
    System.out.println(e.getLocalizedMessage());
    System.exit(1);
}
return null;
}
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [AddAttachmentsToSet](#)
 - [AddCommunicationToCase](#)
 - [CreateCase](#)
 - [DescribeAttachment](#)
 - [DescribeCases](#)
 - [DescribeCommunications](#)
 - [DescribeServices](#)
 - [DescribeSeverityLevels](#)
 - [ResolveCase](#)

使用SDK适用于 Java 2.x 的 Systems Manager 示例

以下代码示例向您展示了如何使用与 Systems Manager AWS SDK for Java 2.x 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

开始使用

Hello Systems Manager

以下代码示例展示了如何开始使用 Systems Manager。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。


```
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);
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [listThings](#)”中的。

主题

- [操作](#)
- [场景](#)

操作

CreateDocument

以下代码示例演示如何使用 CreateDocument。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```

/**
 * 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();
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateDocument](#)”中的。

CreateMaintenanceWindow

以下代码示例演示如何使用 CreateMaintenanceWindow。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```

/**
 * 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];
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateMaintenanceWindow](#)”中的。

CreateOpsItem

以下代码示例演示如何使用 CreateOpsItem。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

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

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [CreateOpsItem](#)”中的。

DeleteDocument

以下代码示例演示如何使用 DeleteDocument。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**
 * 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;
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteDocument](#)”中的。

DeleteMaintenanceWindow

以下代码示例演示如何使用 DeleteMaintenanceWindow。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**
 * 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;
}
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DeleteMaintenanceWindow](#)”中的。

DescribeOpsItems

以下代码示例演示如何使用 DescribeOpsItems。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**
```

```
* 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;
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeOpsItems](#)”中的。

DescribeParameters

以下代码示例演示如何使用 DescribeParameters。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DescribeParameters](#)”中的。

PutParameter

以下代码示例演示如何使用 PutParameter。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
        }
    }
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [PutParameter](#)”中的。

SendCommand

以下代码示例演示如何使用 SendCommand。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**
 * 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];
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [SendCommand](#)”中的。

UpdateMaintenanceWindow

以下代码示例演示如何使用 UpdateMaintenanceWindow。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```

/**
 * 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();
    }
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [UpdateMaintenanceWindow](#)”中的。

UpdateOpsItem

以下代码示例演示如何使用 UpdateOpsItem。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
/**
 * 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;
        }
    }
}

```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [UpdateOpsItem](#)”中的。

场景

开始使用 Systems Manager

以下代码示例说明如何使用 Systems Manager 维护窗口、文档和 OpsItems。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```

import software.amazon.awssdk.services.ssm.model.DocumentAlreadyExistsException;
import software.amazon.awssdk.services.ssm.model.SsmException;

import java.util.Scanner;
public class SSMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        String usage = ""
        Usage:
            <instanceId> <title> <source> <category> <severity>

        Where:
            instanceId - The Amazon EC2 Linux/UNIX instance Id that AWS Systems
Manager uses (ie, i-0149338494ed95f06).
            title - The title of the parameter (default is Disk Space Alert).
            source - The source of the parameter (default is EC2).
    }
}

```

```
        category - The category of the parameter. Valid values are
'Availability', 'Cost', 'Performance', 'Recovery', 'Security' (default is
Performance).
        severity - The severity of the parameter. Severity should be a
number from 1 to 4 (default is 2).
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    Scanner scanner = new Scanner(System.in);
    SSMActions actions = new SSMActions();
    String documentName;
    String windowName;
    String instanceId = args[0];
    String title = args[1];
    String source = args[2];
    String category = args[3];
    String severity = args[4];

    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);

    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 方法的包装器类。

```
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_EAST_1)
            .httpClient(httpClient)
            .overrideConfiguration(overrideConfig)

.credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .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];
}
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [CommandInvocations](#)
 - [CreateDocument](#)
 - [CreateMaintenanceWindow](#)
 - [CreateOpsItem](#)
 - [DeleteMaintenanceWindow](#)
 - [SendCommand](#)
 - [UpdateOpsItem](#)

使用SDK适用于 Java 2.x 的 Amazon Textract 示例

以下代码示例向您展示了如何在 Amazon Textract 中 AWS SDK for Java 2.x 使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)

操作

AnalyzeDocument

以下代码示例演示如何使用 AnalyzeDocument。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [AnalyzeDocument](#)”中的。

DetectDocumentText

以下代码示例演示如何使用 DetectDocumentText。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

从输入文档检测文本。

```
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(textextractClient, sourceDoc);
        textextractClient.close();
    }

    public static void detectDocText(TextextractClient textextractClient, 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 =
textextractClient.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 (TextextractException | FileNotFoundException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

从位于 Amazon S3 桶中的文档检测文本。

```
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(textextractClient, bucketName, docName);
    textextractClient.close();
}

public static void detectDocTextS3(TextextractClient textextractClient, 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 =
textextractClient.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 (TextextractException e) {

        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [DetectDocumentText](#)”中的。

StartDocumentAnalysis

以下代码示例演示如何使用 StartDocumentAnalysis。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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 =
textextractClient.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 =
textextractClient.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 "";
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartDocumentAnalysis](#)”中的。

使用SDK适用于 Java 2.x 的 Amazon Transcribe 示例

以下代码示例向您展示了如何使用 AWS SDK for Java 2.x 与 Amazon Transcribe 配合使用来执行操作和实现常见场景。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过同一服务中调用多个函数来完成特定任务任务的代码示例。

每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关如何在上下文中设置和运行代码的说明。

主题

- [操作](#)
- [场景](#)

操作

ListTranscriptionJobs

以下代码示例演示如何使用 ListTranscriptionJobs。

SDK适用于 Java 2.x

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
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("-----");
    });
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [ListTranscriptionJobs](#)”中的。

StartTranscriptionJob

以下代码示例演示如何使用 StartTranscriptionJob。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

```
public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[])
        throws URISyntaxException, ExecutionException, InterruptedException,
LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
            .credentialsProvider(getCredentials())
            .region(REGION)
            .build();

        CompletableFuture<Void> result =
client.startStreamTranscription(getRequest(16_000),
```

```
        new AudioStreamPublisher(getStreamFromMic()),
        getResponseHandler());

    result.get();
    client.close();
}

private static InputStream getStreamFromMic() throws LineUnavailableException {

    // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
    int sampleRate = 16000;
    AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
    DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

    if (!AudioSystem.isLineSupported(info)) {
        System.out.println("Line not supported");
        System.exit(0);
    }

    TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
    line.open(format);
    line.start();

    InputStream audioStream = new AudioInputStream(line);
    return audioStream;
}

private static AwsCredentialsProvider getCredentials() {
    return DefaultCredentialsProvider.create();
}

private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
    return StartStreamTranscriptionRequest.builder()
        .languageCode(LanguageCode.EN_US.toString())
        .mediaEncoding(MediaEncoding.PCM)
        .mediaSampleRateHertz(mediaSampleRateHertz)
        .build();
}

private static StartStreamTranscriptionResponseHandler getResponseHandler() {
    return StartStreamTranscriptionResponseHandler.builder()
        .onResponse(r -> {
            System.out.println("Received Initial response");
        });
}
```



```

    })
    .onError(e -> {
        System.out.println(e.getMessage());
        StringWriter sw = new StringWriter();
        e.printStackTrace(new PrintWriter(sw));
        System.out.println("Error Occurred: " + sw.toString());
    })
    .onComplete(() -> {
        System.out.println("=== All records stream successfully ===");
    })
    .subscriber(event -> {
        List<Result> results = ((TranscriptEvent)
event).transcript().results();
        if (results.size() > 0) {
            if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

System.out.println(results.get(0).alternatives().get(0).transcript());
            }
        }
    })
    .build();
}

private InputStream getStreamFromFile(String audioFileName) {
    try {
        File inputFile = new
File(getClass().getClassLoader().getResource(audioFileName).getFile());
        InputStream audioStream = new FileInputStream(inputFile);
        return audioStream;
    } catch (FileNotFoundException e) {
        throw new RuntimeException(e);
    }
}

private static class AudioStreamPublisher implements Publisher<AudioStream> {
    private final InputStream inputStream;
    private static Subscription currentSubscription;

    private AudioStreamPublisher(InputStream inputStream) {
        this.inputStream = inputStream;
    }

    @Override

```

```
public void subscribe(Subscriber<? super AudioStream> s) {

    if (this.currentSubscription == null) {
        this.currentSubscription = new SubscriptionImpl(s, inputStream);
    } else {
        this.currentSubscription.cancel();
        this.currentSubscription = new SubscriptionImpl(s, inputStream);
    }
    s.onSubscribe(currentSubscription);
}

}

public static class SubscriptionImpl implements Subscription {
    private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
    private final Subscriber<? super AudioStream> subscriber;
    private final InputStream inputStream;
    private ExecutorService executor = Executors.newFixedThreadPool(1);
    private AtomicLong demand = new AtomicLong(0);

    SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
{
        this.subscriber = s;
        this.inputStream = inputStream;
    }

    @Override
    public void request(long n) {
        if (n <= 0) {
            subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
        }

        demand.getAndAdd(n);

        executor.submit(() -> {
            try {
                do {
                    ByteBuffer audioBuffer = getNextEvent();
                    if (audioBuffer.remaining() > 0) {
                        AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                        subscriber.onNext(audioEvent);
                    } else {
                        subscriber.onComplete();
                    }
                } while (demand.get() > 0);
            } catch (Exception e) {
                subscriber.onError(e);
            }
        });
    }
}
```

```
                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();
}
}
}
```

- 有关API详细信息，请参阅“AWS SDK for Java 2.x API参考 [StartTranscriptionJob](#)”中的。

场景

转录音频并获取任务数据

以下代码示例展示了如何：

- 使用 Amazon Transcribe 开始转录作业。
- 等待作业完成。
- 获取笔URI录的存储位置。

有关更多信息，请参阅 [Amazon Transcribe 入门](#)。

SDK适用于 Java 2.x

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整实例，了解如何进行设置和运行。

转录PCM文件。

```
/**
 * 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();
}
}
```

转录来自计算机麦克风的流音频。

```
public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String args[])
```



```
        throws URISyntaxException, ExecutionException, InterruptedException,
LineUnavailableException {

    client = TranscribeStreamingAsyncClient.builder()
        .credentialsProvider(getCredentials())
        .region(REGION)
        .build();

    CompletableFuture<Void> result =
client.startStreamTranscription(getRequest(16_000),
    new AudioStreamPublisher(getStreamFromMic()),
    getResponseHandler());

    result.get();
    client.close();
}

private static InputStream getStreamFromMic() throws LineUnavailableException {

    // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
    int sampleRate = 16000;
    AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
    DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

    if (!AudioSystem.isLineSupported(info)) {
        System.out.println("Line not supported");
        System.exit(0);
    }

    TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
    line.open(format);
    line.start();

    InputStream audioStream = new AudioInputStream(line);
    return audioStream;
}

private static AwsCredentialsProvider getCredentials() {
    return DefaultCredentialsProvider.create();
}

private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
    return StartStreamTranscriptionRequest.builder()
```

```

        .languageCode(LanguageCode.EN_US.toString())
        .mediaEncoding(MediaEncoding.PCM)
        .mediaSampleRateHertz(mediaSampleRateHertz)
        .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
            .onResponse(r -> {
                System.out.println("Received Initial response");
            })
            .onError(e -> {
                System.out.println(e.getMessage());
                StringWriter sw = new StringWriter();
                e.printStackTrace(new PrintWriter(sw));
                System.out.println("Error Occurred: " + sw.toString());
            })
            .onComplete(() -> {
                System.out.println("=== All records stream successfully ===");
            })
            .subscriber(event -> {
                List<Result> results = ((TranscriptEvent)
event).transcript().results();
                if (results.size() > 0) {
                    if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

                        System.out.println(results.get(0).alternatives().get(0).transcript());
                    }
                }
            })
            .build();
    }

    private InputStream getStreamFromFile(String audioFileName) {
        try {
            File inputFile = new
File(getClass().getClassLoader().getResource(audioFileName).getFile());
            InputStream audioStream = new FileInputStream(inputFile);
            return audioStream;
        } catch (FileNotFoundException e) {
            throw new RuntimeException(e);
        }
    }
}

```

```
private static class AudioStreamPublisher implements Publisher<AudioStream> {
    private final InputStream inputStream;
    private static Subscription currentSubscription;

    private AudioStreamPublisher(InputStream inputStream) {
        this.inputStream = inputStream;
    }

    @Override
    public void subscribe(Subscriber<? super AudioStream> s) {

        if (this.currentSubscription == null) {
            this.currentSubscription = new SubscriptionImpl(s, inputStream);
        } else {
            this.currentSubscription.cancel();
            this.currentSubscription = new SubscriptionImpl(s, inputStream);
        }
        s.onSubscribe(currentSubscription);
    }
}

public static class SubscriptionImpl implements Subscription {
    private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
    private final Subscriber<? super AudioStream> subscriber;
    private final InputStream inputStream;
    private ExecutorService executor = Executors.newFixedThreadPool(1);
    private AtomicLong demand = new AtomicLong(0);

    SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
    {
        this.subscriber = s;
        this.inputStream = inputStream;
    }

    @Override
    public void request(long n) {
        if (n <= 0) {
            subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
        }

        demand.getAndAdd(n);
    }
}
```

```
        executor.submit(() -> {
            try {
                do {
                    ByteBuffer audioBuffer = getNextEvent();
                    if (audioBuffer.remaining() > 0) {
                        AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                        subscriber.onNext(audioEvent);
                    } else {
                        subscriber.onComplete();
                        break;
                    }
                } while (demand.decrementAndGet() > 0);
            } catch (Exception e) {
                subscriber.onError(e);
            }
        });
    }

    @Override
    public void cancel() {
        executor.shutdown();
    }

    private ByteBuffer getNextEvent() {
        ByteBuffer audioBuffer = null;
        byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

        int len = 0;
        try {
            len = inputStream.read(audioBytes);

            if (len <= 0) {
                audioBuffer = ByteBuffer.allocate(0);
            } else {
                audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
            }
        } catch (IOException e) {
            throw new UncheckedIOException(e);
        }

        return audioBuffer;
    }
}
```

```
private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
    return AudioEvent.builder()
        .audioChunk(SdkBytes.fromByteBuffer(bb))
        .build();
}
}
```

- 有关API详细信息，请参阅“参AWS SDK for Java 2.x API考”中的以下主题。
 - [GetTranscriptionJob](#)
 - [StartTranscriptionJob](#)

使用SDK适用于 Java 2.x 的跨服务示例

以下示例应用程序使用 AWS SDK for Java 2.x 来跨多个 AWS 服务工作。

跨服务示例以高级体验为目标，以便您开始构建应用程序。

示例

- [构建应用程序以将数据提交到 DynamoDB 表](#)
- [创建 Amazon Lex 聊天机器人来吸引您的网站访客](#)
- [构建转换消息的发布和订阅应用程序](#)
- [使用 Amazon 创建用于发送和检索消息的 Web 应用程序 SQS](#)
- [创建照片资产管理应用程序，让用户能够使用标签管理照片](#)
- [创建 Web 应用程序来跟踪 DynamoDB 数据](#)
- [创建 Amazon Redshift 项目追踪器](#)
- [创建 Aurora Serverless 工作项跟踪器](#)
- [创建用于分析客户反馈和合成音频的应用程序](#)
- [使用 Amazon Rekognition PPE 在图像中检测 AWS SDK](#)
- [使用 Amazon Rekognition 检测图像中的物体 AWS SDK](#)
- [使用 Amazon Rekognition 检测视频中的人物和物体 AWS SDK](#)
- [使用监控亚马逊 DynamoDB 的性能 AWS SDK](#)
- [使用API网关调用 Lambda 函数](#)

- [使用 Step Functions 调用 Lambda 函数](#)
- [使用计划的事件调用 Lambda 函数](#)

构建应用程序以将数据提交到 DynamoDB 表

SDK适用于 Java 2.x

演示如何创建动态 Web 应用程序，该应用程序使用 Amazon DynamoDB API Java 提交数据并使用亚马逊简单通知服务 Java 发送短信。API

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例[GitHub](#)。

本示例中使用的服务

- DynamoDB
- 亚马逊 SNS

创建 Amazon Lex 聊天机器人来吸引您的网站访客

SDK适用于 Java 2.x

演示如何使用 Amazon Lex API 在网络应用程序中创建聊天机器人以吸引您的网站访问者。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例[GitHub](#)。

本示例中使用的服务

- Amazon Comprehend
- Amazon Lex
- Amazon Translate

构建转换消息的发布和订阅应用程序

SDK适用于 Java 2.x

演示如何使用 Amazon 简单通知服务 Java API 创建具有订阅和发布功能的 Web 应用程序。此外，此示例应用程序还会转换消息。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例[GitHub](#)。

有关如何设置和运行使用 Java Async 的示例的完整源代码和说明API，请参阅上的[GitHub](#)完整示例。

本示例中使用的服务

- 亚马逊 SNS
- Amazon Translate

使用 Amazon 创建用于发送和检索消息的 Web 应用程序 SQS

SDK适用于 Java 2.x

演示如何使用 Amazon 开发SQSAPI用于发送和检索消息RESTAPI的 Spring。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例[GitHub](#)。

本示例中使用的服务

- Amazon Comprehend
- 亚马逊 SQS

创建照片资产管理应用程序，让用户能够使用标签管理照片

SDK适用于 Java 2.x

演示如何开发照片资产管理应用程序，该应用程序使用 Amazon Rekognition 检测图像中的标签并将其存储以供日后检索。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例 [GitHub](#)。

要深入了解这个例子的起源，请参阅 [AWS 社区](#)上的博文。

本示例中使用的服务

- API 网关
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3

- 亚马逊 SNS

创建 Web 应用程序来跟踪 DynamoDB 数据

SDK适用于 Java 2.x

演示如何使用 Amazon Dynamo API DB 创建用于跟踪 DynamoDB 工作数据的动态 Web 应用程序。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例[GitHub](#)。

本示例中使用的服务

- DynamoDB
- 亚马逊 SES

创建 Amazon Redshift 项目追踪器

SDK适用于 Java 2.x

展示如何创建 Web 应用程序来跟踪与报告存储与 Amazon Redshift 数据库的工作项。

有关如何设置查询 Amazon Redshift 数据RESTAPI的 Spring 以及供 React 应用程序使用的完整源代码和说明，请参阅上的完整示例。[GitHub](#)

本示例中使用的服务

- Amazon Redshift
- 亚马逊 SES

创建 Aurora Serverless 工作项跟踪器

SDK适用于 Java 2.x

演示如何创建用于跟踪和报告存储在 Amazon RDS 数据库中的工作项的 Web 应用程序。

有关如何设置查询 Amazon Aurora Serverless 数据RESTAPI的 Spring 以及供 React 应用程序使用的完整源代码和说明，请参阅上的[GitHub](#)完整示例。

有关完整的源代码以及如何设置和运行使用示例的说明 JDBC API，请参阅上的完整示例[GitHub](#)。

本示例中使用的服务

- Aurora
- 亚马逊 RDS
- 亚马逊RDS数据服务
- 亚马逊 SES

创建用于分析客户反馈和合成音频的应用程序

SDK适用于 Java 2.x

此示例应用程序可分析并存储客户反馈卡。具体来说，它满足了纽约市一家虚构酒店的需求。酒店以实体意见卡的形式收集来自不同语种的客人的反馈。该反馈通过 Web 客户端上传到应用程序中。意见卡图片上传后，将执行以下步骤：

- 使用 Amazon Textract 从图片中提取文本。
- Amazon Comprehend 确定所提取文本的情绪及其语言。
- 使用 Amazon Translate 将所提取文本翻译为英语。
- Amazon Polly 根据所提取文本合成音频文件。

完整的应用程序可使用 AWS CDK 进行部署。有关源代码和部署说明，请参阅中的项目 [GitHub](#)。

本示例中使用的服务

- Amazon Comprehend
- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

使用 Amazon Rekognition PPE 在图像中检测 AWS SDK

SDK适用于 Java 2.x

演示如何创建使用个人防护设备检测图像的 AWS Lambda 功能。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例[GitHub](#)。

本示例中使用的服务

- DynamoDB
- Amazon Rekognition
- Amazon S3
- 亚马逊 SES

使用 Amazon Rekognition 检测图像中的物体 AWS SDK

SDK适用于 Java 2.x

演示如何使用 Amazon API Rekognition Java 创建一款应用程序，该应用程序使用 Amazon Rekognition 按类别识别位于亚马逊简单存储服务 (Amazon S3) Simple S3 存储桶中的图像中的对象。该应用程序使用亚马逊简单电子邮件服务 (AmazonSES) 向管理员发送一封包含结果的电子邮件通知。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例[GitHub](#)。

本示例中使用的服务

- Amazon Rekognition
- Amazon S3
- 亚马逊 SES

使用 Amazon Rekognition 检测视频中的人物和物体 AWS SDK

SDK适用于 Java 2.x

演示如何使用 Amazon API Rekognition Java 创建应用程序来检测位于亚马逊简单存储服务 (Amazon S3) 存储桶中的视频中的人脸和物体。该应用程序使用亚马逊简单电子邮件服务 (AmazonSES) 向管理员发送一封包含结果的电子邮件通知。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例[GitHub](#)。

本示例中使用的服务

- Amazon Rekognition
- Amazon S3
- 亚马逊 SES

使用监控亚马逊 DynamoDB 的性能 AWS SDK

SDK适用于 Java 2.x

此示例说明如何配置 Java 应用程序，以监控 DynamoDB 的性能。应用程序将指标数据发送到您可以监控性能 CloudWatch 的地方。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例[GitHub](#)。

本示例中使用的服务

- CloudWatch
- DynamoDB

使用API网关调用 Lambda 函数

SDK适用于 Java 2.x

演示如何使用 Lambda Java 运行时创建 AWS Lambda 函数。API此示例调用不同的 AWS 服务来执行特定的用例。此示例演示如何创建由 Amazon API Gateway 调用的 Lambda 函数，该函数会扫描亚马逊 DynamoDB 表中是否有工作周年纪念日，并使用亚马逊简单通知服务 (AmazonSNS) 向您的员工发送一条短信，祝贺他们在一周年之日向他们表示祝贺。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例[GitHub](#)。

本示例中使用的服务

- API 网关
- DynamoDB
- Lambda
- 亚马逊 SNS

使用 Step Functions 调用 Lambda 函数

SDK适用于 Java 2.x

演示如何使用 AWS Step Functions 和创建 AWS 无服务器工作流程。AWS SDK for Java 2.x每个工作流程步骤都是使用 AWS Lambda 函数实现的。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例[GitHub](#)。

本示例中使用的服务

- DynamoDB
- Lambda
- 亚马逊 SES
- Step Functions

使用计划的事件调用 Lambda 函数

SDK适用于 Java 2.x

演示如何创建调用函数的 Amazon EventBridge 计划事件。AWS Lambda 配置 EventBridge 为使用 cron 表达式来调度 Lambda 函数的调用时间。在此示例中，您将使用 Lambda Java 运行时创建一个 Lambda 函数。API 此示例调用不同的 AWS 服务来执行特定的用例。此示例展示了如何创建一个应用程序，在其一周年纪念日时向员工发送移动短信表示祝贺。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例[GitHub](#)。

本示例中使用的服务

- DynamoDB
- EventBridge
- Lambda
- 亚马逊 SNS

安全性 AWS SDK for Java

云安全性一直是 Amazon Web Services (AWS) 的重中之重。作为 AWS 客户，您将从专为满足大多数安全敏感型企业的要求而打造的数据中心和网络架构中受益。安全是双方 AWS 的共同责任。[责任共担模式](#)将其描述为云的安全性和云中的安全性。

云安全 — AWS 负责保护运行 AWS 云中提供的所有服务的基础架构，并为您提供可以安全使用的服务。我们的安全责任是重中之重 AWS，作为[AWS 合规计划](#)的一部分，第三方审计师定期测试和验证我们安全的有效性。

云端安全 — 您的责任由您使用的 AWS 服务以及其他因素决定，包括数据的敏感性、组织的要求以及适用的法律和法规。

本 AWS 产品或服务通过其支持的特定 Amazon Web Services (AWS) 服务遵循[分担责任模式](#)。有关 AWS 服务安全信息，请参阅[AWS 服务安全文档页面](#)和[合规计划合 AWS 规工作范围内的AWS 服务](#)。

主题

- [AWS SDK for Java 2.x 中的数据保护](#)
- [TLS在 Java 版中SDK使用](#)
- [Identity and Access Management](#)
- [此 AWS 产品或服务的合规性验证](#)
- [本 AWS 产品或服务的弹性](#)
- [本 AWS 产品或服务的基础设施安全](#)

AWS SDK for Java 2.x 中的数据保护

分 AWS [担责任模型](#)适用于中的数据保护 AWS SDK for Java。如本模型所述 AWS，负责保护运行所有内容的全球基础架构 AWS Cloud。您负责维护对托管在此基础架构上的内容的控制。您还负责您所使用的 AWS 服务的安全配置和管理任务。有关数据隐私的更多信息，请参阅[数据隐私FAQ](#)。有关欧洲数据保护的信息，请参阅[责任AWS 共担模型和AWS安全GDPR](#)博客上的博客文章。

出于数据保护目的，我们建议您保护 AWS 账户 凭据并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置个人用户。这样，每个用户只获得履行其工作职责所需的权限。我们还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证 (MFA)。
- 使用SSL/TLS与 AWS 资源通信。我们需要 TLS 1.2，建议使用 TLS 1.3。

- 使用API进行设置和用户活动记录 AWS CloudTrail。
- 使用 AWS 加密解决方案以及其中的所有默认安全控件 AWS 服务。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保護存储在 Amazon S3 中的敏感数据。
- 如果您在 AWS 通过命令行界面或访问时需要 FIPS 140-2 经过验证的加密模块API，请使用端点。FIPS有关可用FIPS端点的更多信息，请参阅[联邦信息处理标准 \(FIPS\) 140-2](#)。

我们强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括当你 AWS 服务 使用SDK控制台、API、或 For Java 或其他 AWS CLI版本时 AWS SDKs。在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您URL向外部服务器提供，我们强烈建议您不要在中包含凭据信息，URL以验证您对该服务器的请求。

TLS在 Java 版中SDK使用

AWS SDK for Java 使用其底层 Java 平台的TLS功能。在本主题中，我们将展示使用 [Amazon Corrett JDK o 17 使用的开放式实现的](#) 示例。

要使用 AWS 服务，底层JDK必须支持最低版本 TLS 1.2，但建议使用 TLS 1.3。

用户应查阅他们使用的 Java 平台的文档，SDK以了解默认情况下启用了哪些TLS版本以及如何启用和禁用特定TLS版本。

如何查看TLS版本信息

使用 OpenJDK，以下代码演示了[SSLContext](#)如何使用来打印支持哪些TLS/SSL版本。

```
System.out.println(Arrays.toString(SSLContext.getDefault().getSupportedSSLParameters().getProtocols()));
```

例如，Amazon Corretto 17 (JDKOpen) 会生成以下输出。

```
[TLSv1.3, TLSv1.2, TLSv1.1, TLSv1, SSLv3, SSLv2Hello]
```

要查看SSL握手的操作以及使用的版本，可以使用系统属性 `java x.net.debug TLS ug`。

例如，运行使用的 Java 应用程序TLS。

```
java app.jar -Djavax.net.debug=ssl:handshake
```

应用程序记录的SSL握手操作类似于以下内容。

```
...
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",
...

```

强制使用最低TLS版本

f SDK or Java 始终首选平台和服务支持的最新TLS版本。如果您想强制使用特定的最低TLS版本，请查阅您的 Java 平台的文档。

对于JDK基于开放的JVMs，您可以使用系统属性`jdk.tls.client.protocols`。

例如，如果您希望应用程序中的SDK服务客户端使用 TLS 1.2（即使 TLS 1.3 可用），请提供以下系统属性。

```
java app.jar -Djdk.tls.client.protocols=TLSv1.2
```

AWS API端点升级到 TLS 1.2

有关 AWS API端点迁移到 TLS 1.2 的最低版本的信息，请参阅此[博客文章](#)。

Identity and Access Management

AWS Identity and Access Management (IAM) AWS 服务 可以帮助管理员安全地控制对 AWS 资源的访问权限。IAM管理员控制谁可以通过身份验证（登录）和授权（拥有权限）使用 AWS 资源。IAM无需支付额外费用即可使用。AWS 服务

主题

- [受众](#)

- [使用身份进行身份验证](#)
- [使用策略管理访问](#)
- [如何 AWS 服务 使用 IAM](#)
- [对 AWS 身份和访问进行故障排除](#)

受众

你使用 AWS Identity and Access Management (IAM) 的方式会有所不同，具体取决于你所做的工作 AWS。

服务用户-如果您 AWS 服务 曾经完成工作，则您的管理员会为您提供所需的凭证和权限。当你使用更多 AWS 功能来完成工作时，你可能需要额外的权限。了解如何管理访问权限有助于您向管理员请求适合的权限。如果您无法访问中的功能 AWS，请参阅[对 AWS 身份和访问进行故障排除](#)或 AWS 服务 您正在使用的用户指南。

服务管理员-如果您负责公司的 AWS 资源，则可能拥有完全访问权限 AWS。您的工作是确定您的服务用户应访问哪些 AWS 功能和资源。然后，您必须向IAM管理员提交更改服务用户权限的请求。查看此页面上的信息以了解的基本概念IAM。要详细了解贵公司如何IAM与配合使用 AWS，请参阅 AWS 服务 您正在使用的用户指南。

IAM管理员-如果您是IAM管理员，则可能需要详细了解如何编写用于管理访问权限的策略 AWS。要查看可在中使用的 AWS 基于身份的策略示例IAM，请参阅 AWS 服务 您正在使用的用户指南。

使用身份进行身份验证

身份验证是您 AWS 使用身份凭证登录的方式。您必须以 AWS 账户根用户、IAM用户身份或通过担任IAM角色进行身份验证（登录 AWS）。

您可以使用通过身份源提供的凭据以 AWS 联合身份登录。AWS IAM Identity Center（IAM身份中心）用户、贵公司的单点登录身份验证以及您的 Google 或 Facebook 凭据就是联合身份的示例。当您以联合身份登录时，您的管理员之前使用IAM角色设置了联合身份。当你使用联合访问 AWS 时，你就是在间接扮演一个角色。

根据您的用户类型，您可以登录 AWS Management Console 或 AWS 访问门户。有关登录的更多信息 AWS，请参阅《AWS 登录 用户指南》[中的如何登录到您 AWS 账户的](#)。

如果您 AWS 以编程方式访问，则会 AWS 提供软件开发套件 (SDK) 和命令行接口 (CLI)，以便使用您的凭据对请求进行加密签名。如果您不使用 AWS 工具，则必须自己签署请求。有关使用推荐的方法自行签署请求的更多信息，请参阅 [《IAM用户指南》中的对 AWS API请求进行签名](#)。

无论使用何种身份验证方法，您可能需要提供其他安全信息。例如，AWS 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅用户指南中的[多重身份验证](#)和AWS IAM Identity Center 用户指南 [AWS中的使用多因素身份验证 \(MFA\)](#)。IAM

AWS 账户 root 用户

创建时 AWS 账户，首先要有一个登录身份，该身份可以完全访问账户中的所有资源 AWS 服务 和资源。此身份被称为 AWS 账户 root 用户，使用您创建账户时使用的电子邮件地址和密码登录即可访问该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关需要您以 root 用户身份登录的任务的完整列表，请参阅《用户指南》中的“[需要根用户凭据的IAM任务](#)”。

联合身份

作为最佳实践，要求人类用户（包括需要管理员访问权限的用户）使用与身份提供商的联合身份验证 AWS 服务 通过临时证书进行访问。

联合身份是指您的企业用户目录、Web 身份提供商、Identity Center 目录中的用户，或者任何使用 AWS 服务 通过身份源提供的凭据进行访问的用户。AWS Directory Service当联合身份访问时 AWS 账户，他们将扮演角色，角色提供临时证书。

要集中管理访问权限，建议您使用 AWS IAM Identity Center。您可以在 Identity Center 中创建用户和群组，也可以连接并同步到您自己的身份源中的一组用户和群组，以便在您的所有 AWS 账户和应用程序中使用。有关IAM身份中心的信息，请参阅[什么是IAM身份中心？](#)在《AWS IAM Identity Center 用户指南》中。

IAM 用户和组

[IAM用户](#)是您内部 AWS 账户 对个人或应用程序具有特定权限的身份。在可能的情况下，我们建议使用临时证书，而不是创建拥有密码和访问密钥等长期凭证的IAM用户。但是，如果您有需要IAM用户长期凭证的特定用例，我们建议您轮换访问密钥。有关更多信息，请参阅《IAM用户指南》中的[定期轮换需要长期凭证的用例的访问密钥](#)。

[IAM群组](#)是指定IAM用户集合的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可以拥有一个名为的群组，IAMAdmins并授予该群组管理IAM资源的权限。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅《[IAM用户指南](#)》中的[何时创建IAM用户（而不是角色）](#)。

IAM 角色

[IAM角色](#)是您内部具有特定权限 AWS 账户 的身份。它与IAM用户类似，但与特定人员无关。您可以 AWS Management Console 通过[切换IAM角色在中临时扮演角色](#)。您可以通过调用 AWS CLI 或 AWS API操作或使用自定义操作来代入角色URL。有关使用角色的方法的更多信息，请参阅《IAM用户指南》中的[使用IAM角色](#)。

IAM具有临时证书的角色在以下情况下很有用：

- 联合用户访问 – 要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关用于联合身份验证的角色的信息，请参阅《IAM用户指南》中的[为第三方身份提供商创建角色](#)。如果您使用 IAM Identity Center，则需要配置权限集。为了控制您的身份在进行身份验证后可以访问的内容，IAM Identity Center 会将权限集关联到中的IAM角色。有关权限集的信息，请参阅《AWS IAM Identity Center 用户指南》中的[权限集](#)。
- 临时IAM用户权限-IAM 用户或角色可以代入一个IAM角色，为特定任务临时获得不同的权限。
- 跨账户访问-您可以使用IAM角色允许其他账户中的某人（受信任的委托人）访问您账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些资源 AWS 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解角色和基于资源的跨账户访问策略之间的区别，请参阅IAM用户指南[IAM中的跨账户资源访问权限](#)。
- 跨服务访问 — 有些 AWS 服务 使用其他 AWS 服务服务中的功能。例如，当您在服务中拨打电话时，该服务通常会在 Amazon 中运行应用程序EC2或在 Amazon S3 中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。
 - 转发访问会话 (FAS)-当您使用IAM用户或角色在中执行操作时 AWS，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS使用调用委托人的权限 AWS 服务以及 AWS 服务 向下游服务发出请求的请求。FAS只有当服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出请求。在这种情况下，您必须具有执行这两个操作的权限。有关提出FAS请求时的政策详情，请参阅[转发访问会话](#)。
- 服务角色-服务[IAM角色](#)是服务代替您执行操作的角色。IAM管理员可以在内部创建、修改和删除服务角色IAM。有关更多信息，请参阅《IAM用户指南》AWS 服务中的[创建角色以向委派权限](#)。
- 服务相关角色-服务相关角色是一种链接到的服务角色。AWS 服务服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM管理员可以查看但不能编辑服务相关角色的权限。
- 在 Amazon 上运行的应用程序 EC2 — 您可以使用IAM角色管理在EC2实例上运行并发出 AWS CLI 或 AWS API请求的应用程序的临时证书。这比在EC2实例中存储访问密钥更可取。要为EC2实例分配 AWS 角色并使其可供其所有应用程序使用，您需要创建一个附加到该实例的实例配置文件。实例

配置文件包含该角色，并允许在EC2实例上运行的程序获得临时证书。有关更多信息，请参阅IAM用户指南中的[使用IAM角色向在 Amazon EC2 实例上运行的应用程序授予权限](#)。

要了解是使用IAM角色还是使用IAM用户，请参阅[《用户指南》中的何时创建IAM角色（而不是IAM用户）](#)。

使用策略管理访问

您可以 AWS 通过创建策略并将其附加到 AWS 身份或资源来控制中的访问权限。策略是其中的一个对象 AWS，当与身份或资源关联时，它会定义其权限。AWS 在委托人（用户、root 用户或角色会话）发出请求时评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略都以JSON文档 AWS 形式存储在中。有关JSON策略文档结构和内容的更多信息，请参阅[《IAM用户指南》中的JSON策略概述](#)。

管理员可以使用 AWS JSON策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

默认情况下，用户和角色没有权限。要授予用户对其所需资源执行操作的权限，IAM管理员可以创建IAM策略。然后，管理员可以将IAM策略添加到角色中，用户可以代入这些角色。

IAM无论您使用何种方法执行操作，策略都会定义该操作的权限。例如，假设您有一个允许 `iam:GetRole` 操作的策略。拥有该策略的用户可以从 AWS Management Console AWS CLI、或获取角色信息 AWS API。

基于身份的策略

基于身份的策略是可以附加到身份（例如IAM用户、用户组或角色）的JSON权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅IAM用户指南中的[创建IAM策略](#)。

基于身份的策略可以进一步归类为内联策略或托管策略。内联策略直接嵌入单个用户、组或角色中。托管策略是独立的策略，您可以将其附加到中的多个用户、群组和角色 AWS 账户。托管策略包括 AWS 托管策略和客户托管策略。要了解如何在托管策略或内联策略之间进行选择，请参阅《IAM用户指南》中的[在托管策略和内联策略之间进行选择](#)。

基于资源的策略

基于资源的JSON策略是您附加到资源的策略文档。基于资源的策略的示例包括IAM角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资

源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中[指定主体](#)。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略IAM中使用 AWS 托管策略。

访问控制列表 (ACLs)

访问控制列表 (ACLs) 控制哪些委托人 (账户成员、用户或角色) 有权访问资源。ACLs与基于资源的策略类似，尽管它们不使用JSON策略文档格式。

Amazon S3 AWS WAF、和亚马逊VPC就是支持的服务示例ACLs。要了解更多信息ACLs，请参阅《亚马逊简单存储服务开发者指南》中的[访问控制列表 \(ACL\) 概述](#)。

其他策略类型

AWS 支持其他不太常见的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界-权限边界是一项高级功能，您可以在其中设置基于身份的策略可以向IAM实体 (IAM用户或角色) 授予的最大权限。您可为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边界的交集。在 Principal 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅《IAM用户指南》中的[IAM实体的权限边界](#)。
- 服务控制策略 (SCPs)-SCPs 是为中的组织或组织单位 (OU) 指定最大权限的JSON策略 AWS Organizations。AWS Organizations 是一项用于对您的企业拥有的多 AWS 账户 项进行分组和集中管理的服务。如果您启用组织中的所有功能，则可以将服务控制策略 (SCPs) 应用于您的任何或所有帐户。对成员账户中的实体 (包括每个实体) 的权限进行了SCP限制 AWS 账户根用户。有关 Organizations 和的更多信息SCPs，[请参阅《AWS Organizations 用户指南》中的SCPs工作原理](#)。
- 会话策略 – 会话策略是当您以编程方式为角色或联合用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅《IAM用户指南》中的[会话策略](#)。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解在涉及多种策略类型时如何 AWS 确定是否允许请求，请参阅IAM用户指南中的[策略评估逻辑](#)。

如何 AWS 服务 使用 IAM

要全面了解如何 AWS 服务 使用大多数IAM功能，请参阅《IAM用户指南》IAM[中与之配合使用的AWS 服务](#)。

要了解如何 AWS 服务 使用特定的IAM，请参阅相关服务《用户指南》的安全部分。

对 AWS 身份和访问进行故障排除

使用以下信息来帮助您诊断和修复在使用 AWS 和时可能遇到的常见问题IAM。

主题

- [我无权在以下位置执行操作 AWS](#)
- [我无权执行 iam : PassRole](#)
- [我想允许我以外的人 AWS 账户 访问我的 AWS 资源](#)

我无权在以下位置执行操作 AWS

如果您收到错误提示，表明您无权执行某个操作，则您必须更新策略以允许执行该操作。

当mateojacksonIAM用户尝试使用控制台查看虚构`my-example-widget`资源的详细信息但没有虚构权限时，就会出现以下示例错误。aws:`GetWidget`

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
aws: GetWidget on resource: my-example-widget
```

在此情况下，必须更新 mateojackson 用户的策略，以允许使用 aws:`GetWidget` 操作访问 `my-example-widget` 资源。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我无权执行 iam : PassRole

如果您收到一个错误，表明您无权执行 iam:PassRole 操作，则必须更新策略以允许您将角色传递给 AWS。

有些 AWS 服务 允许您将现有角色传递给该服务，而不是创建新的服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为的IAM用户marymajor尝试使用控制台在中执行操作时，会出现以下示例错误 AWS。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

在这种情况下，必须更新 Mary 的策略以允许她执行 iam:PassRole 操作。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我想允许我以外的人 AWS 账户 访问我的 AWS 资源

您可以创建一个角色，以便其他账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以担任角色。对于支持基于资源的策略或访问控制列表 (ACLs) 的服务，您可以使用这些策略向人们授予访问您的资源的权限。

要了解更多信息，请参阅以下内容：

- 要了解是否 AWS 支持这些功能，请参阅[如何 AWS 服务 使用 IAM](#)。
- 要了解如何提供对您拥有的资源的[访问权限](#)，请参阅《IAM用户指南》中的[AWS 账户 向其他IAM用户 提供访问权限](#)。AWS 账户
- 要了解如何向第三方提供对您的资源的[访问权限 AWS 账户](#)，请参阅IAM用户指南中的[向第三方提供访问权限](#)。AWS 账户
- 要了解如何通过联合身份验证提供访问权限，请参阅《用户指南》中的[向经过外部身份验证的用户提供访问权限 \(联合身份验证 \)](#)。IAM
- 要了解使用角色和基于资源的策略进行跨账户访问的区别，请参阅IAM用户指南[IAM中的跨账户资源访问权限](#)。

此 AWS 产品或服务的合规性验证

要了解是否属于特定合规计划的范围，请参阅AWS 服务“[按合规计划划分的范围](#)”，然后选择您感兴趣的合规计划。AWS 服务 有关一般信息，请参阅[AWS 合规计划AWS](#)。

您可以使用下载第三方审计报告 AWS Artifact。有关更多信息，请参阅中的“[下载报告](#)”中的“[AWS Artifact](#)”。

您在使用 AWS 服务 时的合规责任取决于您的数据的敏感性、贵公司的合规目标以及适用的法律和法规。AWS 提供了以下资源来帮助实现合规性：

- [安全与合规性快速入门指南](#) — 这些部署指南讨论了架构注意事项，并提供了在这些基础上 AWS 部署以安全性和合规性为重点的基准环境的步骤。
- [在 Amazon Web Services 上进行HIPAA安全与合规架构](#) — 本白皮书描述了各公司如何使用 AWS 来创建HIPAA符合条件的应用程序。

Note

并非所有 AWS 服务 人都有HIPAA资格。有关更多信息，请参阅 [《HIPAA符合条件的服务参考》](#)。

- [AWS 合规资源AWS](#) — 此工作簿和指南集可能适用于您所在的行业和所在地区。
- [AWS 客户合规指南](#) — 从合规角度了解责任共担模式。这些指南总结了保护的最佳实践，AWS 服务并将指南映射到跨多个框架（包括美国国家标准与技术研究院 (NIST)、支付卡行业安全标准委员会 (PCI) 和国际标准化组织 (ISO)）的安全控制。
- [使用AWS Config 开发人员指南中的规则评估资源](#) — 该 AWS Config 服务评估您的资源配置在多大程度上符合内部实践、行业指导方针和法规。
- [AWS Security Hub](#)— 这 AWS 服务 可以全面了解您的安全状态 AWS。Security Hub 通过安全控件评估您的 AWS 资源并检查其是否符合安全行业标准和最佳实践。有关受支持服务及控件的列表，请参阅 [Security Hub 控件参考](#)。
- [Amazon GuardDuty](#) — 它通过监控您的 AWS 账户环境中是否存在可疑和恶意活动，来 AWS 服务 检测您的工作负载、容器和数据面临的潜在威胁。GuardDuty 可以帮助您满足各种合规性要求 PCIDSS，例如满足某些合规性框架规定的入侵检测要求。
- [AWS Audit Manager](#)— 这 AWS 服务 可以帮助您持续审计 AWS 使用情况，从而简化风险管理以及对法规和行业标准的合规性。

本 AWS 产品或服务通过其支持的特定 Amazon Web Services (AWS) 服务遵循[分担责任模式](#)。有关 AWS 服务安全信息，请参阅[AWS 服务安全文档页面](#)和合规[计划合 AWS 规工作范围内的AWS 服务](#)。

本 AWS 产品或服务的弹性

AWS 全球基础设施是围绕 AWS 区域 可用区构建的。

AWS 区域 提供多个物理分隔和隔离的可用区，这些可用区通过低延迟、高吞吐量和高度冗余的网络连接。

利用可用区，您可以设计和操作在可用区之间无中断地自动实现失效转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

有关 AWS 区域和可用区的更多信息，请参阅[AWS 全球基础设施](#)。

本 AWS 产品或服务通过其支持的特定 Amazon Web Services (AWS) 服务遵循[分担责任模式](#)。有关 AWS 服务安全信息，请参阅[AWS 服务安全文档页面](#)和合规[计划符合 AWS 规工作范围内的AWS 服务](#)。

本 AWS 产品或服务的基础设施安全

本 AWS 产品或服务使用托管服务，因此受到 AWS 全球网络安全的保护。有关 AWS 安全服务以及如何 AWS 保护基础设施的信息，请参阅[AWS 云安全](#)。要使用基础设施安全的最佳实践来设计您的 AWS 环境，请参阅 S AWS security Pillar Well-Architected Framework 中的[基础设施保护](#)。

您使用 AWS 已发布的API呼叫通过网络访问本 AWS 产品或服务。客户端必须支持以下内容：

- 传输层安全 (TLS)。我们需要 TLS 1.2，建议使用 TLS 1.3。
- 具有完美前向保密性的密码套件 (PFS)，例如 (Ephemeral Diffie-HellmanPFS) 或 (Elliptic Curve Diffie-Hellman Ephemeral Diffie-Hellman)。ECDHE大多数现代系统 (如 Java 7 及更高版本) 都支持这些模式。

此外，必须使用访问密钥 ID 和与IAM委托人关联的私有访问密钥对请求进行签名。或者，您可以使用[AWS Security Token Service](#) (AWS STS) 生成临时安全凭证来对请求进行签名。

本 AWS 产品或服务通过其支持的特定 Amazon Web Services (AWS) 服务遵循[分担责任模式](#)。有关 AWS 服务安全信息，请参阅[AWS 服务安全文档页面](#)和合规[计划符合 AWS 规工作范围内的AWS 服务](#)。

从 1.x 版迁移到 2.x 版 AWS SDK for Java

AWS SDK for Java 2.x 是对在 Java 8+ 之上构建的 1.x 代码库的重大改写。它包括许多更新，例如改进的一致性、易用性和强制实施的不变性。本部分描述了版本 2.x 中的主要新增功能，并提供有关如何将代码从版本 1.x 迁移到 2.x 的指南。

主题

- [版本 2 中有哪些新功能](#)
- [带示例的迁移 step-by-step 说明](#)
- [Maven ArtifactID 映射的软件包名称](#)
- [AWS SDK for Java 1.x 和 2.x 有什么区别](#)
- [并行使用适用于 Java 的 SDK 1.x 和 2.x 版](#)

版本 2 中有哪些新功能

- 您可以配置自己的 HTTP 客户端。请参阅 [HTTP 传输配置](#)。
- 异步客户端支持非阻塞 I/O 并返回 `CompletableFuture` 对象。请参阅 [异步编程](#)。
- 返回多个页面的操作具有自动分页的响应。这样，您就能够专注于代码对响应执行的操作，而无需检查并获取后续页面。请参阅 [分页](#)。
- AWS Lambda 函数的 SDK 启动时间性能得到改善。请参阅 [SDK 开始时间性能改进](#)。
- 版本 2.x 支持用于创建请求的新快捷方法。

Example

```
dynamoDbClient.putItem(request -> request.tableName(TABLE))
```

要了解有关新功能的更多详细信息并查看特定的代码示例，请参阅本指南的其他部分。

- [Quick Start \(快速入门\)](#)
- [设置](#)
- [AWS SDK for Java 2.x 的代码示例](#)
- [使用 SDK](#)
- [为用户提供安全保障 AWS SDK for Java](#)

带示例的迁移 step-by-step 说明

本节提供了将当前使用适用于 Java 的 SDK v1.x 的应用程序迁移到适用于 Java 2.x 的 SDK 的 step-by-step 指南。第一部分概述了各个步骤，然后是迁移的详细示例。

此处介绍的步骤描述了正常用例的迁移，即应用程序 AWS 服务使用模型驱动的服务客户端进行调用。如果您需要迁移使用更高级别 API（例如 [S3 传输管理器](#) 或 [CloudFront 预签名](#)）的代码，请参阅 [the section called “1.x 和 2.x 之间的差异”](#) 目录下的部分。

此处描述的方法是一个建议。您可以使用其他技术并利用 IDE 的代码编辑功能来获得相同的结果。

步骤概述

1. 首先添加适用于 Java 的 SDK 2.x BOM

通过将适用于 Java SDK 2.x 的 Maven BOM（物料清单）元素添加到您的 POM 文件中，可以确保所需的所有版本 2 依赖项都来自同一个版本。你的 POM 可以同时包含 v1 和 v2 的依赖关系。这允许您以增量方式迁移代码，而不是一次全部更改。

适用于 Java 的 SDK 2.x BOM

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>bom</artifactId>
      <version>2.24.3</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

您可以在 Maven 中央存储库中找到 [最新版本](#)。

2. 在文件中搜索 v1 类的导入语句

通过扫描应用程序中的文件以查找 v1 导入中使用的 service_ID，你会发现所使用的唯一服务 ID。SERVICE_ID 是一个简短、唯一的名称。AWS 服务例如，亚马逊 Co cognitoidentity gnito Identity 的 SERVICE_ID。

3. 从 v1 的导入语句中确定 v2 Maven 的依赖关系

找到所有唯一的 v1 Service_ID 后，您可以通过参考来确定 v2 依赖项的相应的 Maven 工件。[the section called “ArtifactID 映射的包名”](#)

4. 将 v2 依赖项元素添加到 POM 文件中

使用步骤 3 中确定的依赖元素更新 Maven POM 文件。

5. 在 Java 文件中，逐渐将 v1 类更改为 v2 类

在用 v2 类替换 v1 类时，请进行必要的更改以支持 v2 API，例如使用构建器而不是构造函数，以及使用流畅的 getter 和 setter。

6. 从 POM 中移除 v1 Maven 依赖关系，从文件中移除 v1 导入的依赖关系

将代码迁移到使用 v2 类后，请从文件中移除所有剩余的 v1 导入，并从构建文件中移除所有依赖项。

7. 重构代码以使用 v2 API 增强功能

代码成功编译并通过测试后，您可以利用 v2 增强功能，例如使用不同的 HTTP 客户端或分页器来简化代码。此为可选步骤。

迁移示例

在此示例中，我们迁移了一个使用 SDK for Java v1 并可以访问多个应用程序。AWS 服务我们在步骤 5 中详细研究了以下 v1 方法。这是包含八个方法的类中的一个方法，应用程序中有 32 个类。

v1 迁移方法

下面仅列出了从 Java 文件中导入的 v1 软件开发工具包。

```
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. 添加 v2 Maven BOM

将适用于 Java SDK 2.x 的 Maven BOM 以及该部分中的任何其他依赖项一起添加到 POM 中的 `dependencyManagement`。如果您的 POM 文件中有 SDK 版本 1 的 BOM，请暂时将其保留。它将在以后的步骤中删除。

POM 从一开始就进行依赖管理

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.example</groupId>           <!--Existing dependency in POM. -->
      <artifactId>bom</artifactId>
      <version>1.3.4</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
    ...
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-bom</artifactId> <!--Existing v1 BOM dependency. -->
      <version>1.11.1000</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
    ...
    <dependency>
      <groupId>software.amazon.awssdk</groupId> <!--Add v2 BOM dependency. -->
      <artifactId>bom</artifactId>
      <version>2.24.3</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

2. 在文件中搜索 v1 类的导入语句

在应用程序的代码中搜索唯一出现的。import com.amazonaws.services这可以帮助我们确定项目使用的 v1 依赖关系。如果您的应用程序有一个列出了 v1 依赖项的 Maven POM 文件，则可以改用此信息。

在这个例子中，我们使用 [ripgrep\(rg\)](#) 命令来搜索代码库。

从代码库的根目录执行以下ripgrep命令。ripgrep找到导入语句后，会将它们通过管道传送到cutsort、和uniq命令以隔离 Service_ID。

```
rg --no-filename 'import\s+com\.amazonaws\.services' | cut -d '.' -f 4 | sort | uniq
```

对于此应用程序，将以下 Service_ID 记录到控制台。

```
autoscaling
cloudformation
ec2
identitymanagement
```

这表明import语句中使用的以下每个软件包名称至少出现一次。我们的目的有四，各个类名无关紧要。我们只需要找到使用的服务 ID 即可。

```
com.amazonaws.services.autoscaling.*
com.amazonaws.services.cloudformation.*
com.amazonaws.services.ec2.*
com.amazonaws.services.identitymanagement.*
```

3. 从 v1 的导入语句中确定 v2 Maven 的依赖关系

我们从步骤 2 中分离出来的 v1 的 service_ID (例如autoscaling和cloudformation) 在大多数情况下可以映射到相同的 v2 SERVICE_ID。由于 v2 Maven ArtifactID 在大多数情况下都与 SERVICE_ID 匹配，因此你拥有向 POM 文件添加依赖块所需的信息。

下表显示了我们如何确定 v2 依赖关系。

v1 SERVICE_ID 映射到...	v2 SERVICE_ID 映射到...	v2 Maven 依赖关系
包名	包名	
ec2 com.amazonaws.services. ec2 .*	ec2 software.amazon.awssdk.services. ec2 .*	<pre><dependency> <groupId>software.amazon.awssdk</groupId> <artifactId> ec2</artifactId> </dependency></pre>
自动缩放 com.amazonaws.services. autoscaling .*	自动缩放	<pre><dependency></pre>

v1 SERVICE_ID 映射到...	v2 SERVICE_ID 映射到...	v2 Maven 依赖关系
包名	包名	
	software.amazon.awssdk.services. autoscaling .*	<pre><groupId>software. amazon.awssdk</gro upId> <artifactId> autoscali ng </artifactId> </dependency></pre>
cloudformation com.amazonaws.serv ices. cloudform ation .*	cloudformation software.amazon.aw ssdk. cloudform ation .*	<pre><dependency> <groupId>software. amazon.awssdk</gro upId> <artifactId> cloudform ation </artifactId> </dependency></pre>
身份管理* com.amazonaws.serv ices. identitym anagement .*	我* software.amazon.aw ssdk. iam .*	<pre><dependency> <groupId>software. amazon.awssdk</gro upId> <artifactId> iam</ artifactId> </dependency></pre>

* to identitymanagement iam 映射是一个例外，其中 SERVICE_ID 因版本而异。如果 Maven 或 Gradle 无法解析 v2 依赖关系，请参阅了解例外情况。[the section called “ArtifactID 映射的包名”](#)

4. 将 v2 依赖项元素添加到 POM 文件中

在步骤 3 中，我们确定了需要添加到 POM 文件中的四个依赖块。我们不需要添加版本，因为我们在步骤 1 中指定了 BOM。添加导入后，我们的 POM 文件具有以下依赖元素。

```
...
<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. 在 Java 文件中，逐渐将 v1 类更改为 v2 类

在我们正在迁移的方法中，我们看到

- 来自 EC2 服务客户端 `com.amazonaws.services.ec2.AmazonEC2Client`。
- 使用了几个 EC2 模型类。例如 `DescribeInstancesRequest` 和 `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;
}
...

```

我们的目标是用 v2 导入替换所有 v1 导入。我们一次只能上一堂课。

a. 替换导入语句或类名

我们看到该 `describeRunningInstances` 方法的第一个参数是 v1 `AmazonEC2Client` 实例。请执行以下操作之一：

- 将的导入替换为

```
com.amazonaws.services.ec2.AmazonEC2Clientsoftware.amazon.awssdk.services.ec2.
```

然后更改 `AmazonEC2Client` 为 `Ec2Client`。

- 将参数类型更改为 `Ec2Client`，让 IDE 提示我们正确导入。由于客户端名称不同，我们的 IDE 会提示我们导入 `v2` 类，而 `AmazonEC2Client` 且。 `Ec2Client` 如果两个版本中的类名相同，则此方法不起作用。

b. 将 v1 模型类替换为 v2 等效项

在 v2 更改之后 `Ec2Client`，如果我们使用 IDE，则会在以下语句中看到编译错误。

```
result = ec2.describeInstances(request);
```

编译错误源于使用 v1 的实例 `DescribeInstancesRequest` 作为 v2 `Ec2Client` `describeInstances` 方法的参数。要修复此问题，请使用以下替换语句或导入语句。

replace	替换为
<pre>import com.amazonaws.services.ec2.model.DescribeInstancesRequest</pre>	<pre>import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest</pre>

c. 将 v1 构造函数更改为 v2 构建器。

我们仍然看到编译错误，因为 [v2 类上没有构造函数](#)。要修复此问题，请进行以下更改。

更改	到
<pre>final DescribeInstancesRequest request = new DescribeInstancesRequest().withInstanceIds(instanceIdsCopy);</pre>	<pre>final DescribeInstancesRequest request = DescribeInstancesRequest.builder().instanceIds(instanceIdsCopy).build();</pre>

d. 将 v1 `*Result` 响应对象替换为 v `*Response` 2 等效对象

v1 和 v2 之间的一致区别是，v2 [中的所有响应对象都以而不是结 *Response 尾](#)。 `*Result` 将 v1 `DescribeInstancesResult` 导入替换为 v2 导入，。 `DescribeInstancesResponse`

d. 更改 API

以下语句需要进行一些修改。

```
request.setNextToken(result.getNextToken());
```

在 v2 中，[setter 方法](#)不使用 set 或 with。prefix 前缀为的 Getter 方法 get 也已在 Java 2.x 的 SDK 中消失

模型类（例如 request 实例）在 v2 中是不可变的，因此我们需要 DescribeInstancesRequest 使用构建器创建一个新的模型。

在 v2 中，该语句变为以下内容。

```
request = DescribeInstancesRequest.builder()
    .nextToken(result.getNextToken())
    .build();
```

d. 重复直到方法使用 v2 类进行编译

继续执行代码的其余部分。用 v2 导入替换 v1 导入并修复编译错误。如有必要，请参阅 [v2 API 参考](#)和[不同之处参考](#)。

在我们迁移这个单一方法之后，我们有了以下 v2 代码。

```
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;
}
...

```

由于我们在使用八种方法迁移 Java 文件中的单个方法，因此在处理文件时会混合使用 v1 和 v2 导入。我们在执行这些步骤时添加了最后六个 import 语句。

在我们迁移所有代码之后，将不再有 v1 导入语句。

6. 从 POM 中移除 v1 Maven 依赖关系，从文件中移除 v1 导入的依赖关系

迁移文件中的所有 v1 代码后，我们有以下 v2 SDK 导入语句。

```
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;
```

在我们迁移应用程序中的所有文件后，我们不再需要 POM 文件中的 v1 依赖项。从该 `dependencyManagement` 部分中删除 v1 BOM（如果使用）以及所有 v1 依赖项块。

7. 重构代码以使用 v2 API 增强功能

对于我们一直在迁移的片段，我们可以选择使用 v2 分页器，让 SDK 管理基于令牌的更多数据请求。

我们可以将整个 `do` 句替换为以下内容。

```
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 映射的软件包名称

当你将 Maven 或 Gradle 项目从适用于 Java 的 SDK 的 v1 迁移到 v2 时，你需要弄清楚要将哪些依赖项添加到你的构建文件中。[the section called “Step-by-step 指令”](#) (步骤 3) 中描述的方法使用导入语句中的软件包名称作为起点来确定要添加到构建文件中的依赖关系 (如 ArtifactIDs)。

您可以使用本主题中的信息将 v1 软件包名称映射到 v2 ArtifactID。

软件包名称和 Maven ArtifactID 中使用的常见命名惯例

下表显示了软件开发工具包对给定 SERVICE_ID 使用的常见命名约定。SERVICE_ID 是的唯一标识符。AWS 服务例如，亚马逊 S3 服务的 SERVICE_ID 是 Amazon Cognito Identity 的 SERVICE_ID，也是 s3 亚马逊 Cognito cognitoidentity Identity 的服务 ID。

v1 软件包名称 (导入语句)	v1 ArtifactID	v2 ArtifactID	v2 软件包名称 (导入语句)
com.amazonaws.services.servic	aws-java-sdk-服务 ID	服务标识	软件.amazon .awssdk.services.service_ID

亚马逊 Cognito 身份示例 (SERVICE_ID:) *cognitoidentity*

com.amazonaws.servic 认知身份	aws-java-sdk-认知身份	认知身份	软件.amazon .awssdk.services。 认知身份
---------------------------	-------------------	------	-------------------------------------

服务 ID 的差异

在 v1 中

在某些情况下，软件包名称和同一服务的 artifactID 之间的 SERVICE_ID 会有所不同。例如，下表的“CloudWatch 指标”行显示包名称中的 SERVICE_ID，但却 metricscloudwatchmetrics 是 ArtifactID 的 SERVICE_ID。

在 v2 中

软件包名称中使用的 SERVICE_ID 和 artifactID 没有区别。

在 v1 和 v2 之间

对于大多数服务，v2 中的 SERVICE_ID 在软件包名称和 artifactIDs 中都与 v1 的 SERVICE_ID 相同。这方面的一个例子是 cognitoentity SERVICE_ID，如上表所示。但是，软件开发工具包之间有些 Service_ID 有所不同，如下表所示。

v1 中任一列中的粗体 SERVICE_ID 表示它与 v2 中使用的 SERVICE_ID 不同。

服务名称	v1 软件包名称	v1 ArtifactID	v2 ArtifactID	v2 软件包名称
	所有软件包名称都以开头 <code>com.amazonaws.services</code> ，如第一行所示。	如第一行所示，所有 ArtifactID 都包含在标签中。	如第一行所示，所有 ArtifactID 都包含在标签中。	所有软件包名称都以开头 <code>software.amazon.awssdk</code> ，如第一行所示。
API Gateway	<code>com.amazonaws.services.apigateway</code>	<artifactId>aws-java-sdk-api 网关</artifactId>	<artifactId>apigateway</artifactId>	软件.amazon.awssdk.services.apigateway
应用程序注册表	鉴定	鉴定	servicecatalogappregistry	servicecatalogappregistry
Application Discovery	应用程序发现	discovery	应用程序发现	应用程序发现
增强型 AI 运行时	增强空中运行时间	增强空中运行时间	sagemaker a2iRuntime	sagemaker a2iRuntime
Certificate Manager	证书管理器	acm	acm	acm

服务名称	v1 软件包名称	v1 ArtifactID	v2 ArtifactID	v2 软件包名称
CloudControl API	云控制 api	云控制 api	云控制	云控制
CloudSearch	cloudsearch	cloudsearch	cloudsearch	cloudsearch
CloudSearch 域名	云搜索域	云端搜索	云搜索域	云搜索域
CloudWatch 活动	cloudwatcheVent	事件	cloudwatcheVent	cloudwatcheVent
CloudWatch 显然	显而易见	显而易见	evidently	evidently
CloudWatch 日志	日志	日志	云监视日志	云监视日志
CloudWatch 指标	指标	云监视指标	cloudwatch	cloudwatch
CloudWatch 朗姆酒	cloudwatc	cloudwatc	rum	rum
Cognito 身份提供商	cognitoidp	cognitoidp	认知身份提供者	认知身份提供者
Connect 广告系列	连接活动	连接活动	connect广告系列	connect广告系列
Connect 智慧	连接智慧	连接智慧	wisdom	wisdom
Database Migration Service	数据库迁移服务	dms	数据库迁移	数据库迁移
DataZone	datazone	数据区外部	datazone	datazone
DynamoDB	dynamodbv2	dynamodb	dynamodb	dynamodb

服务名称	v1 软件包名称	v1 ArtifactID	v2 ArtifactID	v2 软件包名称
弹性文件系统	弹性文件系统	efs	efs	efs
弹性地图减少	elasticmapreduce	emr	emr	emr
Glue DataBrew	gluedatabrew	gluedatabrew	databrew	databrew
IAM Roles Anywhere	我在任何地方都扮演角色	我在任何地方都扮演角色	rolesanywhere	rolesanywhere
身份管理	身份管理	IAM	IAM	IAM
物联网数据	iotdata	iot	iotdataPlane	iotdataPlane
Kinesis Analytics	kinesisanalytics	kinesis	kinesisanalytics	kinesisanalytics
Kinesis Firehose	运动消防水带	kinesis	firehose	firehose
Kinesis 视频信令频道	kinesis 视频信号频道	kinesis 视频信号频道	kinesis 视频信号	kinesis 视频信号
Lex	lexrunt	lex	lexrunt	lexrunt
注意视力	注意视力	注意视力	lookoutvision	lookoutvision
大型机现代化	大型机现代化	大型机现代化	m2	m2
市场计量	市场计量	市场计量服务	市场计量	市场计量
托管 Grafana	managedgrafana	managedgrafana	grafana	grafana
Mechanical Turk	mturk	机械火鸡请求器	mturk	mturk
Migration Hub Strategy Recommendations	迁移中心策略建议	迁移中心策略建议	迁移中心策略	迁移中心策略
Nimble Studio	灵活的工作室	灵活的工作室	nimble	nimble

服务名称	v1 软件包名称	v1 ArtifactID	v2 ArtifactID	v2 软件包名称
专用 5G	私人 5g	私人 5g	私有网络	私有网络
Prometheus	普罗米修斯	普罗米修斯	放大器	放大器
回收站	回收箱	回收箱	rbin	rbin
Redshift 数据 API	红移数据 api	红移数据 api	红移数据	红移数据
Route 53	route53 个域名	route53	route53 个域名	route53 个域名
Sage Maker 边缘管理	sagemaker edgeManager	sagemaker edgeManager	sagemakeredge	sagemakeredge
安全令牌	安全令牌	sts	sts	sts
服务器迁移	服务器迁移	服务器迁移	sms	sms
简单电子邮件	简单电子邮件	ses	ses	ses
简单电子邮件 V2	simpleemailv2	sesv2	sesv2	sesv2
简单的系统管理	简化系统管理	ssm	ssm	ssm
简单的工作流程	简单的工作流程	简单的工作流程	swf	swf
Step Functions	阶梯函数	阶梯函数	sfn	sfn

AWS SDK for Java 1.x 和 2.x 有什么区别

本节介绍将应用程序从使用 1.x 版本转换为 2.x AWS SDK for Java 版本时需要注意的主要更改。

软件包名称更改

从 SDK 适用于 Java 1.x 的 1.x 版到适用 SDK 于 Java 2.x 版的一个明显变化是软件包名称的更改。Package 名称以 `software.amazon.awssdk` SDK 2.x 开头，而 SDK 1.x 则使用 `com.amazonaws`

这些相同的名称将 Maven 工件从 SDK 1.x 区分为 2.x。SDK 2.x 的 Maven 工件使用 `software.amazon.awssdk` groupId，而 SDK 1.x 则使用 `com.amazonaws` groupId。

有几次，你的代码需要 `com.amazonaws` 依赖于一个原本只使用 SDK 2.x 构件的项目。这方面的一个例子是使用服务器端 AWS Lambda。本指南前面的 [设置 Apache Maven 项目](#) 部分对此进行了介绍。

Note

SDK 1.x 中的几个软件包名称包含 `v2`。在这种情况下，使用 `v2` 通常表示软件包中的代码旨在与相关服务的版本 2 配合使用。

由于软件包的完整名称以开头 `com.amazonaws`，因此这些是 SDK 1.x 组件。SDK 1.x 中这些软件包名称的示例有：

- `com.amazonaws.services.dynamodbv2`
- `com.amazonaws.retry.v2`
- `com.amazonaws.services.apigatewayv2`
- `com.amazonaws.services.simpleemailv2`

将版本 2.x 添加到您的项目

使用 AWS SDK for Java 2.x 时，推荐使用 Maven 来管理依赖关系。要向项目中添加 2.x 版本的组件，请更新您的 `pom.xml` 文件，使其依赖于 SDK

Example

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>bom</artifactId>
      <version>2.16.1</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>

<dependencies>
  <dependency>
```

```
<groupId>software.amazon.awssdk</groupId>
<artifactId>dynamodb</artifactId>
</dependency>
</dependencies>
```

在将项目迁移到 [2.x 版本时 side-by-side](#)，也可以使用 1.x 和 2.x 版本。

不可变 POJOs

客户端和操作的请求和响应对象现在不可变，并且在创建之后不能更改。要重复使用一个请求或响应变量，您必须生成一个要分配给该请求或响应变量的新对象。

Example 在 1.x 中更新请求对象

```
DescribeAlarmsRequest request = new DescribeAlarmsRequest();
DescribeAlarmsResult response = cw.describeAlarms(request);

request.setNextToken(response.getNextToken());
```

Example 在 2.x 中更新请求对象

```
DescribeAlarmsRequest request = DescribeAlarmsRequest.builder().build();
DescribeAlarmsResponse response = cw.describeAlarms(request);

request = DescribeAlarmsRequest.builder()
    .nextToken(response.nextToken())
    .build();
```

设置器和获取器方法

在 AWS SDK for Java 2.x 中，setter 方法名称不包含 set 或 with 前缀。例如，*.withEndpoint() 现为 *.endpoint()。

Getter 方法名称不使用前 get 缀。

Example 在 1.x 中使用设置器方法

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
    .withRegion("us-east-1")
    .build();
```

Example 在 2.x 中使用设置器方法

```
DynamoDbClient client = DynamoDbClient.builder()
    .region(Region.US_EAST_1)
    .build();
```

Example 在 1.x 中使用 getter 方法

```
String token = request.getNextToken();
```

Example 在 2.x 中使用 getter 方法

```
String token = request.nextToken();
```

模型类名

代表服务响应的模型类名以 Response v2 结尾，而不 Result 是 v1 使用的名称。

Example 代表 v1 中响应的类名

```
CreateApiKeyResult
AllocateAddressResult
```

Example 代表 v2 中响应的类名

```
CreateApiKeyResponse
AllocateAddressResponse
```

库和实用工具的迁移状态

SDK适用于 Java 库和实用程序

下表列出了适用于 Java 的库和实用程序 SDK 的迁移状态。

版本 1.12.x 中的名称	版本 2.x 中的名称	自 2.x 的以下版本起
DynamoDBMapper	DynamoDbEnhancedClient	2.12.0
Waiter	Waiter	2.15.0

版本 1.12.x 中的名称	版本 2.x 中的名称	自 2.x 的以下版本起
CloudFrontUrlSigner, CloudFrontCookieSigner	CloudFrontUtilities	2.18.33
TransferManager	S3 TransferManager	2.19.0
EC2元数据客户端	EC2元数据客户端	2.19.29
S3 URI 解析器	S3 URI 解析器	2.20.41
IAM策略生成器	IAM策略生成器	2.20.126
Amazon SQS 客户端缓冲	自动请求批处理	尚未发布
进程侦听程序	进程侦听程序	尚未发布

相关库

下表列出了单独发布但SDK适用于适用于 Java 2.x 的库。

适用于 Java 的 2.x 版本中SDK使用的名称	最低版本
Amazon S3 加密客户端	3.0.0.1
AWS 适用于 DynamoDB 的数据库加密客户端	3.0.0.2

¹适用于 Amazon S3 的加密客户端可通过使用以下 Maven 依赖项获得。

```
<dependency>
  <groupId>software.amazon.encryption.s3</groupId>
  <artifactId>amazon-s3-encryption-client-java</artifactId>
  <version>3.x</version>
</dependency>
```

²使用以下 Maven 依赖项即可使用适用于 DynamoDB 的 AWS 数据库加密客户端。

```
<dependency>
```

```
<groupId>software.amazon.cryptography</groupId>
<artifactId>aws-database-encryption-sdk-dynamodb</artifactId>
<version>3.x</version>
</dependency>
```

库和实用程序的迁移详情

- [S3 传输管理器](#)
- [EC2元数据实用程序](#)
- [CloudFront预签名](#)
- [S3 URI 解析](#)

客户端更改

客户端生成器

您必须使用客户端生成器方法来创建所有客户端。构造函数不再可用。

Example 在版本 1.x 中创建客户端

```
AmazonDynamoDB ddbClient = AmazonDynamoDBClientBuilder.defaultClient();
AmazonDynamoDBClient ddbClient = new AmazonDynamoDBClient();
```

Example 在版本 2.x 中创建客户端

```
DynamoDbClient ddbClient = DynamoDbClient.create();
DynamoDbClient ddbClient = DynamoDbClient.builder().build();
```

客户机类名

所有客户端类名现在完全采用驼峰大小写，且不再使用 Amazon 前缀。这些更改与 AWS CLI中使用的名称保持一致。

Example 1.x 中的类名

```
AmazonDynamoDB
AWSACMPAAsyncClient
```

Example 2.x 中的类名

```
DynamoDbClient
AcmAsyncClient
```

客户机类名变更

1.x 客户端	2.x 客户端
<code>com.amazonaws.services.acmpca.AWSACMPCAAsyncClient</code>	<code>software.amazon.awssdk.services.acm.AcmAsyncClient</code>
<code>com.amazonaws.services.acmpca.AWSACMPCAClient</code>	<code>software.amazon.awssdk.services.acm.AcmClient</code>
<code>com.amazonaws.services.alexaforbusiness.AmazonAlexaForBusinessAsyncClient</code>	<code>software.amazon.awssdk.services.alexaforbusiness.AlexaForBusinessAsyncClient</code>
<code>com.amazonaws.services.alexaforbusiness.AmazonAlexaForBusinessClient</code>	<code>software.amazon.awssdk.services.alexaforbusiness.AlexaForBusinessClient</code>
<code>com.amazonaws.services.apigateway.AmazonApiGatewayAsyncClient</code>	<code>software.amazon.awssdk.services.apigateway.ApiGatewayAsyncClient</code>
<code>com.amazonaws.services.apigateway.AmazonApiGatewayClient</code>	<code>software.amazon.awssdk.services.apigateway.ApiGatewayClient</code>
<code>com.amazonaws.services.applicationautoscaling.AWSApplicationAutoScalingAsyncClient</code>	<code>software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingAsyncClient</code>
<code>com.amazonaws.services.applicationautoscaling.AWSApplicationAutoScalingClient</code>	<code>software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient</code>

1.x 客户端	2.x 客户端
<code>com.amazonaws.services.applicationdiscovery.AWSApplicationDiscoveryAsyncClient</code>	<code>software.amazon.awssdk.services.applicationdiscovery.ApplicationDiscoveryAsyncClient</code>
<code>com.amazonaws.services.applicationdiscovery.AWSApplicationDiscoveryClient</code>	<code>software.amazon.awssdk.services.applicationdiscovery.ApplicationDiscoveryClient</code>
<code>com.amazonaws.services.appstream.AmazonAppStreamAsyncClient</code>	<code>software.amazon.awssdk.services.appstream.AppStreamAsyncClient</code>
<code>com.amazonaws.services.appstream.AmazonAppStreamClient</code>	<code>software.amazon.awssdk.services.appstream.AppStreamClient</code>
<code>com.amazonaws.services.appsync.AWSAppSyncAsyncClient</code>	<code>software.amazon.awssdk.services.appsync.AppSyncAsyncClient</code>
<code>com.amazonaws.services.appsync.AWSAppSyncClient</code>	<code>software.amazon.awssdk.services.appsync.AppSyncClient</code>
<code>com.amazonaws.services.athena.AmazonAthenaAsyncClient</code>	<code>software.amazon.awssdk.services.athena.AthenaAsyncClient</code>
<code>com.amazonaws.services.athena.AmazonAthenaClient</code>	<code>software.amazon.awssdk.services.athena.AthenaClient</code>
<code>com.amazonaws.services.autoscaling.AmazonAutoScalingAsyncClient</code>	<code>software.amazon.awssdk.services.autoscaling.AutoScalingAsyncClient</code>
<code>com.amazonaws.services.autoscaling.AmazonAutoScalingClient</code>	<code>software.amazon.awssdk.services.autoscaling.AutoScalingClient</code>

1.x 客户端	2.x 客户端
<code>com.amazonaws.services.autoscalingplans.AWSAutoScalingPlansAsyncClient</code>	<code>software.amazon.awssdk.services.autoscalingplans.AutoScalingPlansAsyncClient</code>
<code>com.amazonaws.services.autoscalingplans.AWSAutoScalingPlansClient</code>	<code>software.amazon.awssdk.services.autoscalingplans.AutoScalingPlansClient</code>
<code>com.amazonaws.services.batch.AWSBatchAsyncClient</code>	<code>software.amazon.awssdk.services.batch.BatchAsyncClient</code>
<code>com.amazonaws.services.batch.AWSBatchClient</code>	<code>software.amazon.awssdk.services.batch.BatchClient</code>
<code>com.amazonaws.services.budgets.AWSBudgetsAsyncClient</code>	<code>software.amazon.awssdk.services.budgets.BudgetsAsyncClient</code>
<code>com.amazonaws.services.budgets.AWSBudgetsClient</code>	<code>software.amazon.awssdk.services.budgets.BudgetsClient</code>
<code>com.amazonaws.services.certificatemanager.AWSCertificateManagerAsyncClient</code>	<code>software.amazon.awssdk.services.acm.AcmAsyncClient</code>
<code>com.amazonaws.services.certificatemanager.AWSCertificateManagerClient</code>	<code>software.amazon.awssdk.services.acm.AcmClient</code>
<code>com.amazonaws.services.cloud9.AWSCloud9AsyncClient</code>	<code>software.amazon.awssdk.services.cloud9.Cloud9AsyncClient</code>
<code>com.amazonaws.services.cloud9.AWSCloud9Client</code>	<code>software.amazon.awssdk.services.cloud9.Cloud9Client</code>
<code>com.amazonaws.services.clouddirectory.AmazonCloudDirectoryAsyncClient</code>	<code>software.amazon.awssdk.services.clouddirectory.CloudDirectoryAsyncClient</code>

1.x 客户端	2.x 客户端
<code>com.amazonaws.services.clouddirectory.AmazonCloudDirectoryClient</code>	<code>software.amazon.awssdk.services.clouddirectory.CloudDirectoryClient</code>
<code>com.amazonaws.services.cloudformation.AmazonCloudFormationAsyncClient</code>	<code>software.amazon.awssdk.services.cloudformation.CloudFormationAsyncClient</code>
<code>com.amazonaws.services.cloudformation.AmazonCloudFormationClient</code>	<code>software.amazon.awssdk.services.cloudformation.CloudFormationClient</code>
<code>com.amazonaws.services.cloudfront.AmazonCloudFrontAsyncClient</code>	<code>software.amazon.awssdk.services.cloudfront.CloudFrontAsyncClient</code>
<code>com.amazonaws.services.cloudfront.AmazonCloudFrontClient</code>	<code>software.amazon.awssdk.services.cloudfront.CloudFrontClient</code>
<code>com.amazonaws.services.cloudhsm.AWSCloudHSMAsyncClient</code>	<code>software.amazon.awssdk.services.cloudhsm.CloudHsmAsyncClient</code>
<code>com.amazonaws.services.cloudhsm.AWSCloudHSMClient</code>	<code>software.amazon.awssdk.services.cloudhsm.CloudHsmClient</code>
<code>com.amazonaws.services.cloudhsmv2.AWSCloudHSMV2AsyncClient</code>	<code>software.amazon.awssdk.services.cloudhsmv2.CloudHsmV2AsyncClient</code>
<code>com.amazonaws.services.cloudhsmv2.AWSCloudHSMV2Client</code>	<code>software.amazon.awssdk.services.cloudhsmv2.CloudHsmV2Client</code>

1.x 客户端	2.x 客户端
<code>com.amazonaws.services.cloudsearchdomain.AmazonCloudSearchDomainAsyncClient</code>	<code>software.amazon.awssdk.services.cloudsearchdomain.CloudSearchDomainAsyncClient</code>
<code>com.amazonaws.services.cloudsearchdomain.AmazonCloudSearchDomainClient</code>	<code>software.amazon.awssdk.services.cloudsearchdomain.CloudSearchDomainClient</code>
<code>com.amazonaws.services.cloudsearchv2.AmazonCloudSearchAsyncClient</code>	<code>software.amazon.awssdk.services.cloudsearch.CloudSearchAsyncClient</code>
<code>com.amazonaws.services.cloudsearchv2.AmazonCloudSearchClient</code>	<code>software.amazon.awssdk.services.cloudsearch.CloudSearchClient</code>
<code>com.amazonaws.services.cloudtrail.AWSCloudTrailAsyncClient</code>	<code>software.amazon.awssdk.services.cloudtrail.CloudTrailAsyncClient</code>
<code>com.amazonaws.services.cloudtrail.AWSCloudTrailClient</code>	<code>software.amazon.awssdk.services.cloudtrail.CloudTrailClient</code>
<code>com.amazonaws.services.cloudwatch.AmazonCloudWatchAsyncClient</code>	<code>software.amazon.awssdk.services.cloudwatch.CloudWatchAsyncClient</code>
<code>com.amazonaws.services.cloudwatch.AmazonCloudWatchClient</code>	<code>software.amazon.awssdk.services.cloudwatch.CloudWatchClient</code>
<code>com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEventsAsyncClient</code>	<code>software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsAsyncClient</code>

1.x 客户端	2.x 客户端
<code>com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEventsClient</code>	<code>software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient</code>
<code>com.amazonaws.services.codebuild.AWSCodeBuildAsyncClient</code>	<code>software.amazon.awssdk.services.codebuild.CodeBuildAsyncClient</code>
<code>com.amazonaws.services.codebuild.AWSCodeBuildClient</code>	<code>software.amazon.awssdk.services.codebuild.CodeBuildClient</code>
<code>com.amazonaws.services.codecommit.AWSCodeCommitAsyncClient</code>	<code>software.amazon.awssdk.services.codecommit.CodeCommitAsyncClient</code>
<code>com.amazonaws.services.codecommit.AWSCodeCommitClient</code>	<code>software.amazon.awssdk.services.codecommit.CodeCommitClient</code>
<code>com.amazonaws.services.codedeploy.AmazonCodeDeployAsyncClient</code>	<code>software.amazon.awssdk.services.codedeploy.CodeDeployAsyncClient</code>
<code>com.amazonaws.services.codedeploy.AmazonCodeDeployClient</code>	<code>software.amazon.awssdk.services.codedeploy.CodeDeployClient</code>
<code>com.amazonaws.services.codepipeline.AWSCodePipelineAsyncClient</code>	<code>software.amazon.awssdk.services.codepipeline.CodePipelineAsyncClient</code>
<code>com.amazonaws.services.codepipeline.AWSCodePipelineClient</code>	<code>software.amazon.awssdk.services.codepipeline.CodePipelineClient</code>

1.x 客户端	2.x 客户端
<code>com.amazonaws.services.codestar.AWSCodeStarAsyncClient</code>	<code>software.amazon.awssdk.services.codestar.CodeStarAsyncClient</code>
<code>com.amazonaws.services.codestar.AWSCodeStarClient</code>	<code>software.amazon.awssdk.services.codestar.CodeStarClient</code>
<code>com.amazonaws.services.cognitoidentity.AmazonCognitoIdentityAsyncClient</code>	<code>software.amazon.awssdk.services.cognitoidentity.CognitoIdentityAsyncClient</code>
<code>com.amazonaws.services.cognitoidentity.AmazonCognitoIdentityClient</code>	<code>software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient</code>
<code>com.amazonaws.services.cognitoidentityprovider.AWSIdentityProviderAsyncClient</code>	<code>software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderAsyncClient</code>
<code>com.amazonaws.services.cognitoidentityprovider.AWSIdentityProviderClient</code>	<code>software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient</code>
<code>com.amazonaws.services.cognitosync.AmazonCognitoSyncAsyncClient</code>	<code>software.amazon.awssdk.services.cognitosync.CognitoSyncAsyncClient</code>
<code>com.amazonaws.services.cognitosync.AmazonCognitoSyncClient</code>	<code>software.amazon.awssdk.services.cognitosync.CognitoSyncClient</code>
<code>com.amazonaws.services.comprehend.AmazonComprehendAsyncClient</code>	<code>software.amazon.awssdk.services.comprehend.ComprehendAsyncClient</code>

1.x 客户端	2.x 客户端
<code>com.amazonaws.services.comprehend.AmazonComprehendClient</code>	<code>software.amazon.awssdk.services.comprehend.ComprehendClient</code>
<code>com.amazonaws.services.config.AmazonConfigAsyncClient</code>	<code>software.amazon.awssdk.services.config.ConfigAsyncClient</code>
<code>com.amazonaws.services.config.AmazonConfigClient</code>	<code>software.amazon.awssdk.services.config.ConfigClient</code>
<code>com.amazonaws.services.connect.AmazonConnectAsyncClient</code>	<code>software.amazon.awssdk.services.connect.ConnectAsyncClient</code>
<code>com.amazonaws.services.connect.AmazonConnectClient</code>	<code>software.amazon.awssdk.services.connect.ConnectClient</code>
<code>com.amazonaws.services.costandusagereport.AWSCostAndUsageReportAsyncClient</code>	<code>software.amazon.awssdk.services.costandusagereport.CostAndUsageReportAsyncClient</code>
<code>com.amazonaws.services.costandusagereport.AWSCostAndUsageReportClient</code>	<code>software.amazon.awssdk.services.costandusagereport.CostAndUsageReportClient</code>
<code>com.amazonaws.services.costexplorer.AWSCostExplorerAsyncClient</code>	<code>software.amazon.awssdk.services.costexplorer.CostExplorerAsyncClient</code>
<code>com.amazonaws.services.costexplorer.AWSCostExplorerClient</code>	<code>software.amazon.awssdk.services.costexplorer.CostExplorerClient</code>
<code>com.amazonaws.services.databasemigrationservice.AWSDatabaseMigrationServiceAsyncClient</code>	<code>software.amazon.awssdk.services.databasemigration.DatabaseMigrationAsyncClient</code>

1.x 客户端	2.x 客户端
<code>com.amazonaws.services.databasemigrationservice.AWSDatabaseMigrationServiceClient</code>	<code>software.amazon.awssdk.services.databasemigration.DatabaseMigrationClient</code>
<code>com.amazonaws.services.datapipeline.DataPipelineAsyncClient</code>	<code>software.amazon.awssdk.services.datapipeline.DataPipelineAsyncClient</code>
<code>com.amazonaws.services.datapipeline.DataPipelineClient</code>	<code>software.amazon.awssdk.services.datapipeline.DataPipelineAsyncClient</code>
<code>com.amazonaws.services.dax.AmazonDaxAsyncClient</code>	<code>software.amazon.awssdk.services.dax.DaxAsyncClient</code>
<code>com.amazonaws.services.dax.AmazonDaxClient</code>	<code>software.amazon.awssdk.services.dax.DaxClient</code>
<code>com.amazonaws.services.devicefarm.AWSDeviceFarmAsyncClient</code>	<code>software.amazon.awssdk.services.devicefarm.DeviceFarmAsyncClient</code>
<code>com.amazonaws.services.devicefarm.AWSDeviceFarmClient</code>	<code>software.amazon.awssdk.services.devicefarm.DeviceFarmClient</code>
<code>com.amazonaws.services.directconnect.AmazonDirectConnectAsyncClient</code>	<code>software.amazon.awssdk.services.directconnect.DirectConnectAsyncClient</code>
<code>com.amazonaws.services.directconnect.AmazonDirectConnectClient</code>	<code>software.amazon.awssdk.services.directconnect.DirectConnectClient</code>
<code>com.amazonaws.services.directory.AWSDirectoryServiceAsyncClient</code>	<code>software.amazon.awssdk.services.directory.DirectoryAsyncClient</code>

1.x 客户端	2.x 客户端
<code>com.amazonaws.services.directory.AWSDirectoryServiceClient</code>	<code>software.amazon.awssdk.services.directory.DirectoryClient</code>
<code>com.amazonaws.services.dlm.AmazonDLMAsyncClient</code>	<code>software.amazon.awssdk.services.dlm.DlmAsyncClient</code>
<code>com.amazonaws.services.dlm.AmazonDLMClient</code>	<code>software.amazon.awssdk.services.dlm.DlmClient</code>
<code>com.amazonaws.services.dynamodbv2.AmazonDynamoDBAsyncClient</code>	<code>software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient</code>
<code>com.amazonaws.services.dynamodbv2.AmazonDynamoDBClient</code>	<code>software.amazon.awssdk.services.dynamodb.DynamoDbClient</code>
<code>com.amazonaws.services.dynamodbv2.AmazonDynamoDBStreamsAsyncClient</code>	<code>software.amazon.awssdk.services.dynamodb.streams.DynamoDbStreamsAsyncClient</code>
<code>com.amazonaws.services.dynamodbv2.AmazonDynamoDBStreamsClient</code>	<code>software.amazon.awssdk.services.dynamodb.streams.DynamoDbStreamsClient</code>
<code>com.amazonaws.services.ec2.AmazonEC2AsyncClient</code>	<code>software.amazon.awssdk.services.ec2.Ec2AsyncClient</code>
<code>com.amazonaws.services.ec2.AmazonEC2Client</code>	<code>software.amazon.awssdk.services.ec2.Ec2Client</code>
<code>com.amazonaws.services.ecr.AmazonECRAsyncClient</code>	<code>software.amazon.awssdk.services.ecr.EcrAsyncClient</code>
<code>com.amazonaws.services.ecr.AmazonECRClient</code>	<code>software.amazon.awssdk.services.ecr.EcrClient</code>

1.x 客户端	2.x 客户端
<code>com.amazonaws.services.ecs. AmazonECSAsyncClient</code>	<code>software.amazon.awssdk.serv ices.ecs.EcsAsyncClient</code>
<code>com.amazonaws.services.ecs. AmazonECSClient</code>	<code>software.amazon.awssdk.serv ices.ecs.EcsClient</code>
<code>com.amazonaws.services.eks. AmazonEKSAsyncClient</code>	<code>software.amazon.awssdk.serv ices.eks.EksAsyncClient</code>
<code>com.amazonaws.services.eks. AmazonEKSClient</code>	<code>software.amazon.awssdk.serv ices.eks.EksClient</code>
<code>com.amazonaws.services.elas ticache.AmazonElasticCacheAs yncClient</code>	<code>software.amazon.awssdk.serv ices.elasticache.ElastiCach eAsyncClient</code>
<code>com.amazonaws.services.elas ticache.AmazonElasticCacheClient</code>	<code>software.amazon.awssdk.serv ices.elasticache.ElastiCach eClient</code>
<code>com.amazonaws.services.elas ticbeanstalk.AWSElasticBean stalkAsyncClient</code>	<code>software.amazon.awssdk.serv ices.elasticbeanstalk.Elast icBeanstalkAsyncClient</code>
<code>com.amazonaws.services.elas ticbeanstalk.AWSElasticBean stalkClient</code>	<code>software.amazon.awssdk.serv ices.elasticbeanstalk.Elast icBeanstalkClient</code>
<code>com.amazonaws.services.elas ticfilesystem.AmazonElastic FileSystemAsyncClient</code>	<code>software.amazon.awssdk.serv ices.efs.EfsAsyncClient</code>
<code>com.amazonaws.services.elas ticfilesystem.AmazonElastic FileSystemClient</code>	<code>software.amazon.awssdk.serv ices.efs.EfsClient</code>

1.x 客户端	2.x 客户端
<code>com.amazonaws.services.elasticloadbalancing.AmazonElasticLoadBalancingAsyncClient</code>	<code>software.amazon.awssdk.services.elasticloadbalancing.ElasticLoadBalancingAsyncClient</code>
<code>com.amazonaws.services.elasticloadbalancing.AmazonElasticLoadBalancingClient</code>	<code>software.amazon.awssdk.services.elasticloadbalancing.ElasticLoadBalancingClient</code>
<code>com.amazonaws.services.elasticloadbalancingv2.AmazonElasticLoadBalancingAsyncClient</code>	<code>software.amazon.awssdk.services.elasticloadbalancingv2.ElasticLoadBalancingV2AsyncClient</code>
<code>com.amazonaws.services.elasticloadbalancingv2.AmazonElasticLoadBalancingClient</code>	<code>software.amazon.awssdk.services.elasticloadbalancingv2.ElasticLoadBalancingV2Client</code>
<code>com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduceAsyncClient</code>	<code>software.amazon.awssdk.services.emr.EmrAsyncClient</code>
<code>com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduceClient</code>	<code>software.amazon.awssdk.services.emr.EmrClient</code>
<code>com.amazonaws.services.elasticsearch.AWSElasticsearchAsyncClient</code>	<code>software.amazon.awssdk.services.elasticsearch.ElasticsearchAsyncClient</code>
<code>com.amazonaws.services.elasticsearch.AWSElasticsearchClient</code>	<code>software.amazon.awssdk.services.elasticsearch.ElasticsearchClient</code>
<code>com.amazonaws.services.elastictranscoder.AmazonElasticTranscoderAsyncClient</code>	<code>software.amazon.awssdk.services.elastictranscoder.ElasticTranscoderAsyncClient</code>

1.x 客户端	2.x 客户端
<code>com.amazonaws.services.elastictranscoder.AmazonElasticTranscoderClient</code>	<code>software.amazon.awssdk.services.elastictranscoder.ElasticTranscoderClient</code>
<code>com.amazonaws.services.fms.AWSFMSAsyncClient</code>	<code>software.amazon.awssdk.services.fms.FmsAsyncClient</code>
<code>com.amazonaws.services.fms.AWSFMSClient</code>	<code>software.amazon.awssdk.services.fms.FmsClient</code>
<code>com.amazonaws.services.gamelift.AmazonGameLiftAsyncClient</code>	<code>software.amazon.awssdk.services.gamelift.GameLiftAsyncClient</code>
<code>com.amazonaws.services.gamelift.AmazonGameLiftClient</code>	<code>software.amazon.awssdk.services.gamelift.GameLiftClient</code>
<code>com.amazonaws.services.glacier.AmazonGlacierAsyncClient</code>	<code>software.amazon.awssdk.services.glacier.GlacierAsyncClient</code>
<code>com.amazonaws.services.glacier.AmazonGlacierClient</code>	<code>software.amazon.awssdk.services.glacier.GlacierClient</code>
<code>com.amazonaws.services.glue.AWSGlueAsyncClient</code>	<code>software.amazon.awssdk.services.glue.GlueAsyncClient</code>
<code>com.amazonaws.services.glue.AWSGlueClient</code>	<code>software.amazon.awssdk.services.glue.GlueClient</code>
<code>com.amazonaws.services.greengrass.AWSGreengrassAsyncClient</code>	<code>software.amazon.awssdk.services.greengrass.GreengrassAsyncClient</code>
<code>com.amazonaws.services.greengrass.AWSGreengrassClient</code>	<code>software.amazon.awssdk.services.greengrass.GreengrassClient</code>

1.x 客户端	2.x 客户端
<code>com.amazonaws.services.guardduty.AmazonGuardDutyAsyncClient</code>	<code>software.amazon.awssdk.services.guardduty.GuardDutyAsyncClient</code>
<code>com.amazonaws.services.guardduty.AmazonGuardDutyClient</code>	<code>software.amazon.awssdk.services.guardduty.GuardDutyClient</code>
<code>com.amazonaws.services.health.AWSHealthAsyncClient</code>	<code>software.amazon.awssdk.services.health.HealthAsyncClient</code>
<code>com.amazonaws.services.health.AWSHealthClient</code>	<code>software.amazon.awssdk.services.health.HealthClient</code>
<code>com.amazonaws.services.identitymanagement.AmazonIdentityManagementAsyncClient</code>	<code>software.amazon.awssdk.services.iam.IamAsyncClient</code>
<code>com.amazonaws.services.identitymanagement.AmazonIdentityManagementClient</code>	<code>software.amazon.awssdk.services.iam.IamClient</code>
<code>com.amazonaws.services.importexport.AmazonImportExportAsyncClient</code>	<i>Deprecated</i>
<code>com.amazonaws.services.importexport.AmazonImportExportClient</code>	<i>Deprecated</i>
<code>com.amazonaws.services.inspector.AmazonInspectorAsyncClient</code>	<code>software.amazon.awssdk.services.inspector.InspectorAsyncClient</code>
<code>com.amazonaws.services.inspector.AmazonInspectorClient</code>	<code>software.amazon.awssdk.services.inspector.InspectorClient</code>

1.x 客户端	2.x 客户端
<code>com.amazonaws.services.iot.AWSIoTAsyncClient</code>	<code>software.amazon.awssdk.services.iot.IotAsyncClient</code>
<code>com.amazonaws.services.iot.AWSIoTClient</code>	<code>software.amazon.awssdk.services.iot.IotClient</code>
<code>com.amazonaws.services.iot1clickdevices.AWSIoT1ClickDevicesAsyncClient</code>	<code>software.amazon.awssdk.services.iot1clickdevices.Iot1ClickDevicesAsyncClient</code>
<code>com.amazonaws.services.iot1clickdevices.AWSIoT1ClickDevicesClient</code>	<code>software.amazon.awssdk.services.iot1clickdevices.Iot1ClickDevicesClient</code>
<code>com.amazonaws.services.iot1clickprojects.AWSIoT1ClickProjectsAsyncClient</code>	<code>software.amazon.awssdk.services.iot1clickprojects.Iot1ClickProjectsAsyncClient</code>
<code>com.amazonaws.services.iot1clickprojects.AWSIoT1ClickProjectsClient</code>	<code>software.amazon.awssdk.services.iot1clickprojects.Iot1ClickProjectsClient</code>
<code>com.amazonaws.services.iotanalytics.AWSIoTAnalyticsAsyncClient</code>	<code>software.amazon.awssdk.services.iotanalytics.IotAnalyticsAsyncClient</code>
<code>com.amazonaws.services.iotanalytics.AWSIoTAnalyticsClient</code>	<code>software.amazon.awssdk.services.iotanalytics.IotAnalyticsClient</code>
<code>com.amazonaws.services.iotdata.AWSIoTDataAsyncClient</code>	<code>software.amazon.awssdk.services.iotdata.IotDataAsyncClient</code>
<code>com.amazonaws.services.iotdata.AWSIoTDataClient</code>	<code>software.amazon.awssdk.services.iotdata.IotDataClient</code>

1.x 客户端	2.x 客户端
<code>com.amazonaws.services.iotjobsdataplane.AWSIoTJobsDataPlaneAsyncClient</code>	<code>software.amazon.awssdk.services.iotdataplane.IotDataPlaneAsyncClient</code>
<code>com.amazonaws.services.iotjobsdataplane.AWSIoTJobsDataPlaneClient</code>	<code>software.amazon.awssdk.services.iotdataplane.IotDataPlaneClient</code>
<code>com.amazonaws.services.kinesis.AmazonKinesisAsyncClient</code>	<code>software.amazon.awssdk.services.kinesis.KinesisAsyncClient</code>
<code>com.amazonaws.services.kinesis.AmazonKinesisClient</code>	<code>software.amazon.awssdk.services.kinesis.KinesisClient</code>
<code>com.amazonaws.services.kinesisanalytics.AmazonKinesisAnalyticsAsyncClient</code>	<code>software.amazon.awssdk.services.kinesisanalytics.KinesisAnalyticsAsyncClient</code>
<code>com.amazonaws.services.kinesisanalytics.AmazonKinesisAnalyticsClient</code>	<code>software.amazon.awssdk.services.kinesisanalytics.KinesisAnalyticsClient</code>
<code>com.amazonaws.services.kinesisfirehose.AmazonKinesisFirehoseAsyncClient</code>	<code>software.amazon.awssdk.services.firehose.FirehoseAsyncClient</code>
<code>com.amazonaws.services.kinesisfirehose.AmazonKinesisFirehoseClient</code>	<code>software.amazon.awssdk.services.firehose.FirehoseClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoArchivedMediaAsyncClient</code>	<code>software.amazon.awssdk.services.kinesisvideoarchivedmedia.KinesisVideoArchivedMediaAsyncClient</code>

1.x 客户端	2.x 客户端
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoArchivedMediaClient</code>	<code>software.amazon.awssdk.services.kinesisvideoarchivedmedia.KinesisVideoArchivedMediaClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoAsyncClient</code>	<code>software.amazon.awssdk.services.kinesisvideo.KinesisVideoAsyncClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoClient</code>	<code>software.amazon.awssdk.services.kinesisvideo.KinesisVideoClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoMediaAsyncClient</code>	<code>software.amazon.awssdk.services.kinesisvideomedia.KinesisVideoMediaAsyncClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoMediaClient</code>	<code>software.amazon.awssdk.services.kinesisvideomedia.KinesisVideoMediaClient</code>
<code>com.amazonaws.services.kinesisvideo.AmazonKinesisVideoPutMediaClient</code>	不支持
<code>com.amazonaws.services.kms.AWSKMSAsyncClient</code>	<code>software.amazon.awssdk.services.kms.KmsAsyncClient</code>
<code>com.amazonaws.services.kms.AWSKMSClient</code>	<code>software.amazon.awssdk.services.kms.KmsClient</code>
<code>com.amazonaws.services.lambda.AWSLambdaAsyncClient</code>	<code>software.amazon.awssdk.services.lambda.LambdaAsyncClient</code>
<code>com.amazonaws.services.lambda.AWSLambdaClient</code>	<code>software.amazon.awssdk.services.lambda.LambdaClient</code>

1.x 客户端	2.x 客户端
<code>com.amazonaws.services.lexmodelbuilding.AmazonLexModelBuildingAsyncClient</code>	<code>software.amazon.awssdk.services.lexmodelbuilding.LexModelBuildingAsyncClient</code>
<code>com.amazonaws.services.lexmodelbuilding.AmazonLexModelBuildingClient</code>	<code>software.amazon.awssdk.services.lexmodelbuilding.LexModelBuildingClient</code>
<code>com.amazonaws.services.lexruntime.AmazonLexRuntimeAsyncClient</code>	<code>software.amazon.awssdk.services.lexruntime.LexRuntimeAsyncClient</code>
<code>com.amazonaws.services.lexruntime.AmazonLexRuntimeClient</code>	<code>software.amazon.awssdk.services.lexruntime.LexRuntimeClient</code>
<code>com.amazonaws.services.lightsail.AmazonLightsailAsyncClient</code>	<code>software.amazon.awssdk.services.lightsail.LightsailAsyncClient</code>
<code>com.amazonaws.services.lightsail.AmazonLightsailClient</code>	<code>software.amazon.awssdk.services.lightsail.LightsailClient</code>
<code>com.amazonaws.services.logs.AWSLogsAsyncClient</code>	<code>software.amazon.awssdk.services.logs.LogsAsyncClient</code>
<code>com.amazonaws.services.logs.AWSLogsClient</code>	<code>software.amazon.awssdk.services.logs.LogsClient</code>
<code>com.amazonaws.services.machinelearning.AmazonMachineLearningAsyncClient</code>	<code>software.amazon.awssdk.services.machinelearning.MachineLearningAsyncClient</code>
<code>com.amazonaws.services.machinelearning.AmazonMachineLearningClient</code>	<code>software.amazon.awssdk.services.machinelearning.MachineLearningClient</code>

1.x 客户端	2.x 客户端
<code>com.amazonaws.services.macie.AmazonMacieAsyncClient</code>	<code>software.amazon.awssdk.services.macie.MacieAsyncClient</code>
<code>com.amazonaws.services.macie.AmazonMacieClient</code>	<code>software.amazon.awssdk.services.macie.MacieClient</code>
<code>com.amazonaws.services.marketplacecommerceanalytics.AWSMarketplaceCommerceAnalyticsAsyncClient</code>	<code>software.amazon.awssdk.services.marketplacecommerceanalytics.MarketplaceCommerceAnalyticsAsyncClient</code>
<code>com.amazonaws.services.marketplacecommerceanalytics.AWSMarketplaceCommerceAnalyticsClient</code>	<code>software.amazon.awssdk.services.marketplacecommerceanalytics.MarketplaceCommerceAnalyticsClient</code>
<code>com.amazonaws.services.marketplaceentitlement.AWSMarketplaceEntitlementAsyncClient</code>	<code>software.amazon.awssdk.services.marketplaceentitlement.MarketplaceEntitlementAsyncClient</code>
<code>com.amazonaws.services.marketplaceentitlement.AWSMarketplaceEntitlementClient</code>	<code>software.amazon.awssdk.services.marketplaceentitlement.MarketplaceEntitlementClient</code>
<code>com.amazonaws.services.marketplacemetering.AWSMarketplaceMeteringAsyncClient</code>	<code>software.amazon.awssdk.services.marketplacemetering.MarketplaceMeteringAsyncClient</code>
<code>com.amazonaws.services.marketplacemetering.AWSMarketplaceMeteringClient</code>	<code>software.amazon.awssdk.services.marketplacemetering.MarketplaceMeteringClient</code>
<code>com.amazonaws.services.mediaconvert.AWSMediaConvertAsyncClient</code>	<code>software.amazon.awssdk.services.mediaconvert.MediaConvertAsyncClient</code>

1.x 客户端	2.x 客户端
<code>com.amazonaws.services.mediaconvert.AWSMediaConvertClient</code>	<code>software.amazon.awssdk.services.mediaconvert.MediaConvertClient</code>
<code>com.amazonaws.services.medialive.AWSMediaLiveAsyncClient</code>	<code>software.amazon.awssdk.services.medialive.MediaLiveAsyncClient</code>
<code>com.amazonaws.services.medialive.AWSMediaLiveClient</code>	<code>software.amazon.awssdk.services.medialive.MediaLiveClient</code>
<code>com.amazonaws.services.mediapackage.AWSMediaPackageAsyncClient</code>	<code>software.amazon.awssdk.services.mediapackage.MediaPackageAsyncClient</code>
<code>com.amazonaws.services.mediapackage.AWSMediaPackageClient</code>	<code>software.amazon.awssdk.services.mediapackage.MediaPackageClient</code>
<code>com.amazonaws.services.mediastore.AWSMediaStoreAsyncClient</code>	<code>software.amazon.awssdk.services.mediastore.MediaStoreAsyncClient</code>
<code>com.amazonaws.services.mediastore.AWSMediaStoreClient</code>	<code>software.amazon.awssdk.services.mediastore.MediaStoreClient</code>
<code>com.amazonaws.services.mediastoredata.AWSMediaStoreDataAsyncClient</code>	<code>software.amazon.awssdk.services.mediastoredata.MediaStoreDataAsyncClient</code>
<code>com.amazonaws.services.mediastoredata.AWSMediaStoreDataClient</code>	<code>software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient</code>

1.x 客户端	2.x 客户端
<code>com.amazonaws.services.mediataylor.AWSMediaTailorAsyncClient</code>	<code>software.amazon.awssdk.services.mediataylor.MediaTailorAsyncClient</code>
<code>com.amazonaws.services.mediataylor.AWSMediaTailorClient</code>	<code>software.amazon.awssdk.services.mediataylor.MediaTailorClient</code>
<code>com.amazonaws.services.migrationhub.AWSMigrationHubAsyncClient</code>	<code>software.amazon.awssdk.services.migrationhub.MigrationHubAsyncClient</code>
<code>com.amazonaws.services.migrationhub.AWSMigrationHubClient</code>	<code>software.amazon.awssdk.services.migrationhub.MigrationHubClient</code>
<code>com.amazonaws.services.mobile.AWSMobileAsyncClient</code>	<code>software.amazon.awssdk.services.mobile.MobileAsyncClient</code>
<code>com.amazonaws.services.mobile.AWSMobileClient</code>	<code>software.amazon.awssdk.services.mobile.MobileClient</code>
<code>com.amazonaws.services.mq.AmazonMQAsyncClient</code>	<code>software.amazon.awssdk.services.mq.MqAsyncClient</code>
<code>com.amazonaws.services.mq.AmazonMQClient</code>	<code>software.amazon.awssdk.services.mq.MqClient</code>
<code>com.amazonaws.services.mturk.AmazonMTurkAsyncClient</code>	<code>software.amazon.awssdk.services.mturk.MTurkAsyncClient</code>
<code>com.amazonaws.services.mturk.AmazonMTurkClient</code>	<code>software.amazon.awssdk.services.mturk.MTurkClient</code>
<code>com.amazonaws.services.neptune.AmazonNeptuneAsyncClient</code>	<code>software.amazon.awssdk.services.neptune.NeptuneAsyncClient</code>

1.x 客户端	2.x 客户端
<code>com.amazonaws.services.neptune.AmazonNeptuneClient</code>	<code>software.amazon.awssdk.services.neptune.NeptuneClient</code>
<code>com.amazonaws.services.opsworks.AWSOpsWorksAsyncClient</code>	<code>software.amazon.awssdk.services.opsworks.OpsWorksAsyncClient</code>
<code>com.amazonaws.services.opsworks.AWSOpsWorksClient</code>	<code>software.amazon.awssdk.services.opsworks.OpsWorksClient</code>
<code>com.amazonaws.services.opsworkscm.AWSOpsWorksCMAsyncClient</code>	<code>software.amazon.awssdk.services.opsworkscm.OpsWorksCmAsyncClient</code>
<code>com.amazonaws.services.opsworkscm.AWSOpsWorksCMClient</code>	<code>software.amazon.awssdk.services.opsworkscm.OpsWorksCmClient</code>
<code>com.amazonaws.services.organizations.AWSOrganizationsAsyncClient</code>	<code>software.amazon.awssdk.services.organizations.OrganizationsAsyncClient</code>
<code>com.amazonaws.services.organizations.AWSOrganizationsClient</code>	<code>software.amazon.awssdk.services.organizations.OrganizationsClient</code>
<code>com.amazonaws.services.pi.AWSPIAsyncClient</code>	<code>software.amazon.awssdk.services.pi.PiAsyncClient</code>
<code>com.amazonaws.services.pi.AWSPIClient</code>	<code>software.amazon.awssdk.services.pi.PiClient</code>
<code>com.amazonaws.services.pinpoint.AmazonPinpointAsyncClient</code>	<code>software.amazon.awssdk.services.pinpoint.PinpointAsyncClient</code>

1.x 客户端	2.x 客户端
<code>com.amazonaws.services.pinpoint.AmazonPinpointClient</code>	<code>software.amazon.awssdk.services.pinpoint.PinpointClient</code>
<code>com.amazonaws.services.polly.AmazonPollyAsyncClient</code>	<code>software.amazon.awssdk.services.polly.PollyAsyncClient</code>
<code>com.amazonaws.services.polly.AmazonPollyClient</code>	<code>software.amazon.awssdk.services.polly.PollyClient</code>
<code>com.amazonaws.services.pricing.AWS PricingAsyncClient</code>	<code>software.amazon.awssdk.services.pricing.PricingAsyncClient</code>
<code>com.amazonaws.services.pricing.AWS PricingClient</code>	<code>software.amazon.awssdk.services.pricing.PricingClient</code>
<code>com.amazonaws.services.rds.AmazonRDSAsyncClient</code>	<code>software.amazon.awssdk.services.rds.RdsAsyncClient</code>
<code>com.amazonaws.services.rds.AmazonRDSClient</code>	<code>software.amazon.awssdk.services.rds.RdsClient</code>
<code>com.amazonaws.services.redshift.AmazonRedshiftAsyncClient</code>	<code>software.amazon.awssdk.services.redshift.RedshiftAsyncClient</code>
<code>com.amazonaws.services.redshift.AmazonRedshiftClient</code>	<code>software.amazon.awssdk.services.redshift.RedshiftClient</code>
<code>com.amazonaws.services.rekognition.AmazonRekognitionAsyncClient</code>	<code>software.amazon.awssdk.services.rekognition.RekognitionAsyncClient</code>
<code>com.amazonaws.services.rekognition.AmazonRekognitionClient</code>	<code>software.amazon.awssdk.services.rekognition.RekognitionClient</code>

1.x 客户端	2.x 客户端
<code>com.amazonaws.services.resourcegroups.AWSResourceGroupsAsyncClient</code>	<code>software.amazon.awssdk.services.resourcegroups.ResourceGroupsAsyncClient</code>
<code>com.amazonaws.services.resourcegroups.AWSResourceGroupsClient</code>	<code>software.amazon.awssdk.services.resourcegroups.ResourceGroupsClient</code>
<code>com.amazonaws.services.resourcegroupstaggingapi.AWSResourceGroupsTaggingAPIAsyncClient</code>	<code>software.amazon.awssdk.services.resourcegroupstaggingapi.ResourceGroupsTaggingAPIAsyncClient</code>
<code>com.amazonaws.services.resourcegroupstaggingapi.AWSResourceGroupsTaggingAPIClient</code>	<code>software.amazon.awssdk.services.resourcegroupstaggingapi.ResourceGroupsTaggingAPIClient</code>
<code>com.amazonaws.services.route53.AmazonRoute53AsyncClient</code>	<code>software.amazon.awssdk.services.route53.Route53AsyncClient</code>
<code>com.amazonaws.services.route53.AmazonRoute53Client</code>	<code>software.amazon.awssdk.services.route53.Route53Client</code>
<code>com.amazonaws.services.route53domains.AmazonRoute53DomainsAsyncClient</code>	<code>software.amazon.awssdk.services.route53domains.Route53DomainsAsyncClient</code>
<code>com.amazonaws.services.route53domains.AmazonRoute53DomainsClient</code>	<code>software.amazon.awssdk.services.route53domains.Route53DomainsClient</code>
<code>com.amazonaws.services.s3.AmazonS3Client</code>	<code>software.amazon.awssdk.services.s3.S3Client</code>

1.x 客户端	2.x 客户端
<code>com.amazonaws.services.sagemaker.AmazonSageMakerAsyncClient</code>	<code>software.amazon.awssdk.services.sagemaker.SageMakerAsyncClient</code>
<code>com.amazonaws.services.sagemaker.AmazonSageMakerClient</code>	<code>software.amazon.awssdk.services.sagemaker.SageMakerClient</code>
<code>com.amazonaws.services.sagemakerruntime.AmazonSageMakerRuntimeAsyncClient</code>	<code>software.amazon.awssdk.services.sagemakerruntime.SageMakerRuntimeAsyncClient</code>
<code>com.amazonaws.services.sagemakerruntime.AmazonSageMakerRuntimeClient</code>	<code>software.amazon.awssdk.services.sagemakerruntime.SageMakerRuntimeClient</code>
<code>com.amazonaws.services.secretsmanager.AWSSecretsManagerAsyncClient</code>	<code>software.amazon.awssdk.services.secretsmanager.SecretsManagerAsyncClient</code>
<code>com.amazonaws.services.secretsmanager.AWSSecretsManagerClient</code>	<code>software.amazon.awssdk.services.secretsmanager.SecretsManagerClient</code>
<code>com.amazonaws.services.securitytoken.AWSSecurityTokenServiceAsyncClient</code>	<code>software.amazon.awssdk.services.sts.StsAsyncClient</code>
<code>com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClient</code>	<code>software.amazon.awssdk.services.sts.StsClient</code>
<code>com.amazonaws.services.serverlessapplicationrepository.AWSServerlessApplicationRepositoryAsyncClient</code>	<code>software.amazon.awssdk.services.serverlessapplicationrepository.ServerlessApplicationRepositoryAsyncClient</code>

1.x 客户端	2.x 客户端
<code>com.amazonaws.services.serverlessapplicationrepository.AWSServerlessApplicationRepositoryClient</code>	<code>software.amazon.awssdk.services.serverlessapplicationrepository.ServerlessApplicationRepositoryClient</code>
<code>com.amazonaws.services.servermigration.AWSServerMigrationAsyncClient</code>	<code>software.amazon.awssdk.services.sms.SmsAsyncClient</code>
<code>com.amazonaws.services.servermigration.AWSServerMigrationClient</code>	<code>software.amazon.awssdk.services.sms.SmsClient</code>
<code>com.amazonaws.services.servicecatalog.AWSServiceCatalogAsyncClient</code>	<code>software.amazon.awssdk.services.servicecatalog.ServiceCatalogAsyncClient</code>
<code>com.amazonaws.services.servicecatalog.AWSServiceCatalogClient</code>	<code>software.amazon.awssdk.services.servicecatalog.ServiceCatalogClient</code>
<code>com.amazonaws.services.servicediscovery.AWSServiceDiscoveryAsyncClient</code>	<code>software.amazon.awssdk.services.servicediscovery.ServiceDiscoveryAsyncClient</code>
<code>com.amazonaws.services.servicediscovery.AWSServiceDiscoveryClient</code>	<code>software.amazon.awssdk.services.servicediscovery.ServiceDiscoveryClient</code>
<code>com.amazonaws.services.shield.AWSShieldAsyncClient</code>	<code>software.amazon.awssdk.services.shield.ShieldAsyncClient</code>
<code>com.amazonaws.services.shield.AWSShieldClient</code>	<code>software.amazon.awssdk.services.shield.ShieldClient</code>

1.x 客户端	2.x 客户端
<code>com.amazonaws.services.simp ledb.AmazonSimpleDBAsyncClient</code>	<code>software.amazon.awssdk.serv ices.simplesdb.SimpleDbAsync Client</code>
<code>com.amazonaws.services.simp ledb.AmazonSimpleDBClient</code>	<code>software.amazon.awssdk.serv ices.simplesdb.SimpleDbClient</code>
<code>com.amazonaws.services.simp leemail.AmazonSimpleEmailSe rviceAsyncClient</code>	<code>software.amazon.awssdk.serv ices.ses.SesAsyncClient</code>
<code>com.amazonaws.services.simp leemail.AmazonSimpleEmailSe rviceClient</code>	<code>software.amazon.awssdk.serv ices.ses.SesClient</code>
<code>com.amazonaws.services.simp lesystemsmanagement.AWSSimp leSystemsManagementAsyncClient</code>	<code>software.amazon.awssdk.serv ices.ssm.SsmAsyncClient</code>
<code>com.amazonaws.services.simp lesystemsmanagement.AWSSimp leSystemsManagementClient</code>	<code>software.amazon.awssdk.serv ices.ssm.SsmClient</code>
<code>com.amazonaws.services.simp leworkflow.AmazonSimpleWork flowAsyncClient</code>	<code>software.amazon.awssdk.serv ices.swf.SwfAsyncClient</code>
<code>com.amazonaws.services.simp leworkflow.AmazonSimpleWork flowClient</code>	<code>software.amazon.awssdk.serv ices.swf.SwfClient</code>
<code>com.amazonaws.services.snow ball.AmazonSnowballAsyncClient</code>	<code>software.amazon.awssdk.serv ices.snowball.SnowballAsync Client</code>
<code>com.amazonaws.services.snow ball.AmazonSnowballClient</code>	<code>software.amazon.awssdk.serv ices.snowball.SnowballClient</code>

1.x 客户端	2.x 客户端
<code>com.amazonaws.services.sns. AmazonSNSAsyncClient</code>	<code>software.amazon.awssdk.serv ices.sns.SnsAsyncClient</code>
<code>com.amazonaws.services.sns. AmazonSNSClient</code>	<code>software.amazon.awssdk.serv ices.sns.SnsClient</code>
<code>com.amazonaws.services.sqs. AmazonSQSAsyncClient</code>	<code>software.amazon.awssdk.serv ices.sqs.SqsAsyncClient</code>
<code>com.amazonaws.services.sqs. AmazonSQSClient</code>	<code>software.amazon.awssdk.serv ices.sqs.SqsClient</code>
<code>com.amazonaws.services.step functions.AWSStepFunctionsA syncClient</code>	<code>software.amazon.awssdk.serv ices.sfn.SfnAsyncClient</code>
<code>com.amazonaws.services.step functions.AWSStepFunctionsC lient</code>	<code>software.amazon.awssdk.serv ices.sfn.SfnClient</code>
<code>com.amazonaws.services.stor agegateway.AWSStorageGatewa yAsyncClient</code>	<code>software.amazon.awssdk.serv ices.storagegateway.Storage GatewayAsyncClient</code>
<code>com.amazonaws.services.stor agegateway.AWSStorageGatewa yClient</code>	<code>software.amazon.awssdk.serv ices.storagegateway.Storage GatewayClient</code>
<code>com.amazonaws.services.supp ort.AWSSupportAsyncClient</code>	<code>software.amazon.awssdk.serv ices.support.SupportAsyncClient</code>
<code>com.amazonaws.services.supp ort.AWSSupportClient</code>	<code>software.amazon.awssdk.serv ices.support.SupportClient</code>
<code>com.amazonaws.services.tran scribe.AmazonTranscribeAsyn cClient</code>	<code>software.amazon.awssdk.serv ices.transcribe.TranscribeA syncClient</code>

1.x 客户端	2.x 客户端
<code>com.amazonaws.services.transcribe.AmazonTranscribeClient</code>	<code>software.amazon.awssdk.services.transcribe.TranscribeClient</code>
<code>com.amazonaws.services.translate.AmazonTranslateAsyncClient</code>	<code>software.amazon.awssdk.services.translate.TranslateAsyncClient</code>
<code>com.amazonaws.services.translate.AmazonTranslateClient</code>	<code>software.amazon.awssdk.services.translate.TranslateClient</code>
<code>com.amazonaws.services.waf.AWSWAFAsyncClient</code>	<code>software.amazon.awssdk.services.waf.WafAsyncClient</code>
<code>com.amazonaws.services.waf.AWSWAFClient</code>	<code>software.amazon.awssdk.services.waf.WafClient</code>
<code>com.amazonaws.services.waf.AWSWAFRegionalAsyncClient</code>	<code>software.amazon.awssdk.services.waf.regional.WafRegionalAsyncClient</code>
<code>com.amazonaws.services.waf.AWSWAFRegionalClient</code>	<code>software.amazon.awssdk.services.waf.regional.WafRegionalClient</code>
<code>com.amazonaws.services.workdocs.AmazonWorkDocsAsyncClient</code>	<code>software.amazon.awssdk.services.workdocs.WorkDocsAsyncClient</code>
<code>com.amazonaws.services.workdocs.AmazonWorkDocsClient</code>	<code>software.amazon.awssdk.services.workdocs.WorkDocsClient</code>
<code>com.amazonaws.services.workmail.AmazonWorkMailAsyncClient</code>	<code>software.amazon.awssdk.services.workmail.WorkMailAsyncClient</code>

1.x 客户端	2.x 客户端
<code>com.amazonaws.services.workmail.AmazonWorkMailClient</code>	<code>software.amazon.awssdk.services.workmail.WorkMailClient</code>
<code>com.amazonaws.services.workspaces.AmazonWorkspacesAsyncClient</code>	<code>software.amazon.awssdk.services.workspaces.WorkspacesAsyncClient</code>
<code>com.amazonaws.services.workspaces.AmazonWorkspacesClient</code>	<code>software.amazon.awssdk.services.workspaces.WorkspacesClient</code>
<code>com.amazonaws.services.xray.AWSXRayAsyncClient</code>	<code>software.amazon.awssdk.services.xray.XRayAsyncClient</code>
<code>com.amazonaws.services.xray.AWSXRayClient</code>	<code>software.amazon.awssdk.services.xray.XRayClient</code>

默认创建客户端

在版本 2.x 中，对默认的客户端创建逻辑进行了以下更改。

- S3 的默认凭证提供程序链不再包含匿名证书。您必须使用手动指定对 S3 的匿名访问权限 `AnonymousCredentialsProvider`。
- 以下与创建默认客户端相关的环境变量有所不同。

1.x	2.x
<code>AWS_CBOR_DISABLED</code>	<code>CBOR_ENABLED</code>
<code>AWS_ION_BINARY_DISABLE</code>	<code>BINARY_ION_ENABLED</code>

- 以下与创建默认客户机相关的系统属性有所不同。

1.x	2.x
<code>com.amazonaws.sdk.disableEc2Metadata</code>	<code>aws.disableEc2Metadata</code>
<code>com.amazonaws.sdk.ec2MetadataServiceEndpointOverride</code>	<code>aws.ec2MetadataServiceEndpoint</code>
<code>com.amazonaws.sdk.disableCbor</code>	<code>aws.cborEnabled</code>
<code>com.amazonaws.sdk.disableIoBinary</code>	<code>aws.binaryIonEnabled</code>

- 版本 2.x 不支持以下系统属性。

1.x
<code>com.amazonaws.sdk.disableCertChecking</code>
<code>com.amazonaws.sdk.enableDefaultMetrics</code>
<code>com.amazonaws.sdk.enableThrottledRetry</code>
<code>com.amazonaws.regions.RegionUtils.fileOverride</code>
<code>com.amazonaws.regions.RegionUtils.disableRemote</code>
<code>com.amazonaws.services.s3.disableImplicitGlobalClients</code>
<code>com.amazonaws.sdk.enableInRegionOptimizedMode</code>

- 不再支持从自定义 `endpoints.json` 文件加载区域配置。

客户端配置

在 1.x 中，通过在 SDK 客户端或客户端生成器上设置 `ClientConfiguration` 实例来修改客户端配置。在版本 2.x 中，已将客户端配置分为单独的配置类。使用单独的配置类，您可以为异步 HTTP 客户端和同步客户端配置不同的客户端，但仍使用相同的 `ClientOverrideConfiguration` 类。

Example 版本 1.x 中的客户端配置

```
AmazonDynamoDBClientBuilder.standard()
    .withClientConfiguration(clientConfiguration)
    .build()
```

Example 版本 2.x 中的同步客户端配置

```
ProxyConfiguration.Builder proxyConfig = ProxyConfiguration.builder();

ApacheHttpClient.Builder httpClientBuilder =
    ApacheHttpClient.builder()
        .proxyConfiguration(proxyConfig.build());

ClientOverrideConfiguration.Builder overrideConfig =
    ClientOverrideConfiguration.builder();

DynamoDbClient client =
    DynamoDbClient.builder()
        .httpClientBuilder(httpClientBuilder)
        .overrideConfiguration(overrideConfig.build())
        .build();
```

Example 版本 2.x 中的异步客户端配置

```
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 客户

显著的变化

- 在版本 2.x 中，您可以通过使用指定实现来更改在运行时使用的 HTTP 客户端。 `clientBuilder.httpClientBuilder`
- 使用 `clientBuilder.httpClient` 将客户端传递给服务 HTTP 客户端生成器时，如果服务 HTTP 客户端关闭，则默认情况下该客户端不会关闭。这允许您在服务 HTTP 客户端之间共享客户端。
- 异步 HTTP 客户端现在使用非阻塞 IO。
- 有些操作现在使用 HTTP /2 来提高性能。

设置变更

设置	1.x	2.x 同步, Apache	2.x Async, Netty
	<pre>ClientCon figuration clientConfig = new ClientCon figuration()</pre>	<pre>ApacheHtt pClient.B uilder httpClien tBuilder = ApacheHtt pClient.b uilder()</pre>	<pre>NettyNioA syncHttpC lient.Builder httpClient tBuilder = NettyNioA syncHttpC lient.builder()</pre>
最大连接数	<pre>clientCon fig.setMa xConnecti ons(...) clientCon fig.withM axConnect ions(...)</pre>	<pre>httpClien tBuilder. maxConnec tions(...)</pre>	<pre>httpClien tBuilder. maxConcur rency(...)</pre>
连接超时	<pre>clientCon fig.setCo nnectionT imeout(...)</pre>	<pre>httpClien tBuilder. connectio nTimeout(...)</pre>	<pre>httpClien tBuilder. connectio nTimeout(...)</pre>

设置	1.x	2.x 同步, Apache	2.x Async, Netty
	<code>clientConfig.withConnectionTimeout(...)</code>	<code>httpClientBuilder.connectionAcquisitionTimeout(...)</code>	
套接字超时	<code>clientConfig.setSocketTimeout(...)</code> <code>clientConfig.withSocketTimeout(...)</code>	<code>httpClientBuilder.socketTimeout(...)</code>	<code>httpClientBuilder.writeTimeout(...)</code> <code>httpClientBuilder.readTimeout(...)</code>
连接 TTL	<code>clientConfig.setConnectionTTL(...)</code> <code>clientConfig.withConnectionTTL(...)</code>	<code>httpClientBuilder.connectionTimeToLive(...)</code>	<code>httpClientBuilder.connectionTimeToLive(...)</code>
连接最大空闲时间	<code>clientConfig.setConnectionMaxIdleMillis(...)</code> <code>clientConfig.withConnectionMaxIdleMillis(...)</code>	<code>httpClientBuilder.connectionMaxIdleTime(...)</code>	<code>httpClientBuilder.connectionMaxIdleTime(...)</code>

设置	1.x	2.x 同步 , Apache	2.x Async , Netty
处于非活动状态后进行验证	<pre>clientConfig.setValidateAfterInactivityMillis(...) clientConfig.withValidateAfterInactivityMillis(...)</pre>	不支持 (请求功能)	不支持 (请求功能)
本地地址	<pre>clientConfig.setLocalAddress(...) clientConfig.withLocalAddress(...)</pre>	<pre>httpClientBuilder.localAddress(...)</pre>	不支持
已启用“预期继续”	<pre>clientConfig.setUseExpectContinue(...) clientConfig.withUseExpectContinue(...)</pre>	<pre>httpClientBuilder.expectContinueEnabled(...)</pre>	不支持 (请求功能)
连接收割者	<pre>clientConfig.setUseReaper(...) clientConfig.withReaper(...)</pre>	<pre>httpClientBuilder.useIdleConnectionReaper(...)</pre>	<pre>httpClientBuilder.useIdleConnectionReaper(...)</pre>

设置	1.x	2.x 同步, Apache	2.x Async, Netty
	<pre>AmazonDynamoDBClientBuilder .standard() .withClientConfiguration(clientConfiguration) .build()</pre>	<pre>DynamoDbClient.builder() .httpClientBuilder(httpClientBuilder) .build()</pre>	<pre>DynamoDbAsyncClient.builder() .httpClientBuilder(httpClientBuilder) .build()</pre>

HTTP客户端代理

设置	1.x	2.x 同步, Apache	2.x Async, Netty
	<pre>ClientConfiguration clientConfig = new ClientConfiguration()</pre>	<pre>ProxyConfiguration .Builder proxyConfig = ProxyConfiguration .builder()</pre>	<pre>ProxyConfiguration .Builder proxyConfig = ProxyConfiguration .builder()</pre>
代理主机	<pre>clientConfig.setProxyHost(...) clientConfig.withProxyHost(...)</pre>	<pre>proxyConfig.endpoint(...)</pre>	<pre>proxyConfig.host(...)</pre>
代理端口	<pre>clientConfig.setProxyPort(...) clientConfig.withProxyPort(...)</pre>	<pre>proxyConfig.endpoint(...)</pre>	<pre>proxyConfig.port(...)</pre>

设置	1.x	2.x 同步, Apache	2.x Async, Netty
		代理端口 嵌入在 endpoint	
代理用户名	<pre>clientConfig.setProxyUsername(...) clientConfig.withProxyUsername(...)</pre>	<pre>proxyConfig.username(...)</pre>	<pre>proxyConfig.username(...)</pre>
代理密码	<pre>clientConfig.setProxyPassword(...) clientConfig.withProxyPassword(...)</pre>	<pre>proxyConfig.password(...)</pre>	<pre>proxyConfig.password(...)</pre>
代理域	<pre>clientConfig.setProxyDomain(...) clientConfig.withProxyDomain(...)</pre>	<pre>proxyConfig.ntlmDomain(...)</pre>	不支持 (请求功能)
代理工作站	<pre>clientConfig.setProxyWorkspace(...) clientConfig.withProxyWorkstation(...)</pre>	<pre>proxyConfig.ntlmWorkstation(...)</pre>	不支持 (请求功能)

设置	1.x	2.x 同步 , Apache	2.x Async , Netty
代理身份验证方法	<pre>clientConfig.setProxyAuthenticationMethods(...) clientConfig.withProxyAuthenticationMethods(...)</pre>	<u>不支持</u>	不支持 (请求功能)
抢占式基本代理身份验证	<pre>clientConfig.setPreemptiveBasicProxyAuth(...) clientConfig.withPreemptiveBasicProxyAuth(...)</pre>	<pre>proxyConfig.preemptiveBasicAuthenticationEnabled(...)</pre>	不支持 (请求功能)
非代理主机	<pre>clientConfig.setNonProxyHosts(...) clientConfig.withNonProxyHosts(...)</pre>	<pre>proxyConfig.nonProxyHosts(...)</pre>	<pre>proxyConfig.nonProxyHosts(...)</pre>

设置	1.x	2.x 同步, Apache	2.x Async, Netty
禁用套接字代理	<pre>clientConfig.setDisableSocketProxy(...) clientConfig.withDisableSocketProxy(...)</pre>	不支持 (请求功能)	不支持 (请求功能)
	<pre>AmazonDynamoDBClientBuilder .standard() .withClientConfiguration(clientConfiguration) .build()</pre>	<pre>httpClientBuilder.proxyConfiguration(proxyConfiguration).build()</pre>	<pre>httpClientBuilder.proxyConfiguration(proxyConfiguration).build()</pre>

客户机覆盖

设置	1.x	2.x
	<pre>ClientConfiguration clientConfig = new ClientConfiguration()</pre>	<pre>ClientOverrideConfiguration.Builder overrideConfig = ClientOverrideConfiguration.builder()</pre>
用户代理前缀	<pre>clientConfig.setUserAgentPrefix(...) clientConfig.withUserAgentPrefix(...)</pre>	<pre>overrideConfig.advancedOption(SdkAdvancedClientOption.USER_AGENT_PREFIX, ...)</pre>

设置	1.x	2.x
用户代理后缀	<pre>clientConfig.setUserAgentSuffix(...) clientConfig.withUserAgentSuffix(...)</pre>	<pre>overrideConfig.advancedOption(SdkAdvancedClientOption.USER_AGENT_SUFFIX, ...)</pre>
Signer	<pre>clientConfig.setSignerOverride(...) clientConfig.withSignerOverride(...)</pre>	<pre>overrideConfig.advancedOption(SdkAdvancedClientOption.SIGNER, ...)</pre>
其他标题	<pre>clientConfig.addHeader(...) clientConfig.withHeader(...)</pre>	<pre>overrideConfig.putHeader(...)</pre>
请求超时	<pre>clientConfig.setRequestTimeout(...) clientConfig.withRequestTimeout(...)</pre>	<pre>overrideConfig.apiCallAttemptTimeout(...)</pre>
客户端执行超时	<pre>clientConfig.setClientExecutionTimeout(...) clientConfig.withClientExecutionTimeout(...)</pre>	<pre>overrideConfig.apiCallTimeout(...)</pre>
使用 Gzip	<pre>clientConfig.setUseGzip(...) clientConfig.withGzip(...)</pre>	不支持 (请求功能)

设置	1.x	2.x
套接字缓冲区大小提示	<pre>clientConfig.setSocketBufferSizeHints(...) clientConfig.withSocketBufferSizeHints(...)</pre>	不支持 (请求功能)
缓存响应元数据	<pre>clientConfig.setCacheResponseMetadata(...) clientConfig.withCacheResponseMetadata(...)</pre>	不支持 (请求功能)
响应元数据缓存大小	<pre>clientConfig.setResponseMetadataCacheSize(...) clientConfig.withResponseMetadataCacheSize(...)</pre>	不支持 (请求功能)
DNS解析器	<pre>clientConfig.setDnsResolver(...) clientConfig.withDnsResolver(...)</pre>	不支持 (请求功能)
TCP保持活力	<pre>clientConfig.setUseTcpKeepAlive(...) clientConfig.withTcpKeepAlive(...)</pre>	<p>此选项现在位于HTTP客户机配置中</p> <ul style="list-style-type: none"> - <code>ApacheHttpClient.builder().tcpKeepAlive(true)</code> - <code>NettyNioAsyncHttpClient.builder().tcpKeepAlive(true)</code>

设置	1.x	2.x
安全随机	<pre>clientConfig.setSecureRandom(...) clientConfig.withSecureRandom(...)</pre>	不支持 (请求功能)
	<pre>AmazonDynamoDBClientBuilder.standard() .withClientConfiguration(clientConfiguration) .build()</pre>	<pre>DynamoDbClient.builder() .httpClientBuilder(httpClientBuilder) .build()</pre>

客户端重试次数

设置	1.x	2.x
	<pre>ClientConfiguration clientConfig = new ClientConfiguration()</pre>	<pre>RetryPolicy.Builder retryPolicy = RetryPolicy.builder()</pre>
最大错误重试次数	<pre>clientConfig.setMaxErrorRetry(...) clientConfig.withMaxErrorRetry(...)</pre>	<pre>retryPolicy.numRetries(...)</pre>
使用受限制的重试次数	<pre>clientConfig.setUseThrottleRetries(...) clientConfig.withUseThrottleRetries(...)</pre>	不支持
限制前的最大连续重试次数	<pre>clientConfig.setMaxConsecutiveRetrie</pre>	不支持

设置	1.x	2.x
	<pre>sBeforeThrottling(...) clientConfig.withMaxCo nsecutiveRetriesBe foreThrottling(...)</pre>	
	<pre>AmazonDynamoDBClie ntBuilder.standard() .withClientConfigu ration(clientConfi guration) .build()</pre>	<pre>DynamoDbClient.bui lder() .httpClientBuilder (httpClientBuilder) .build()</pre>

异步客户端

设置	1.x	2.x
		<pre>ClientAsyncConfigu ration.Builder asyncConfig = ClientAsyncConfigu ration.builder()</pre>
执行程序	<pre>AmazonDynamoDBAsyn cClientBuilder.sta ndard() .withExecutorFacto ry(...) .build()</pre>	<pre>asyncConfig.advanc edOption(SdkAdvancedAsynCli entOption.FUTURE_ COMPLETION_EXECUTO R, ...)</pre>
		<pre>DynamoDbAsynClien t.builder() .asyncConfiguratio n(asyncConfig) .build()</pre>

其他客户机变更

1.x 中的以下 ClientConfiguration 选项在 2.x 中已更改，没有直接 SDK 等效项。

设置	1.x	等同于 2.x
协议	<pre>clientConfig.setProtocol(Protocol.HTTP) clientConfig.withProtocol(Protocol.HTTP)</pre>	<p>协议设置为 HTTPS 默认设置。要修改设置，请在客户端生成器上指定协议设置 HTTP 端点：</p> <pre>clientBuilder.endpointOverride(URI.create("http://..."))</pre>

凭证提供程序更改

本部分提供了 AWS SDK for Java 1.x 与 2.x 版之间的凭证提供程序类和方法名称更改的映射。

显著差异

- 在版本 2.x 中，默认凭证提供程序会在加载环境变量之前先加载系统属性。有关更多信息，请参阅[使用凭证](#)。
- 构造函数方法已被替换为 create 或 builder 方法。

Example

```
DefaultCredentialsProvider.create();
```

- 默认情况下，不再设置异步刷新。您必须使用凭证提供程序的 builder 指定它。

Example

```
ContainerCredentialsProvider provider = ContainerCredentialsProvider.builder()
    .asyncCredentialUpdateEnabled(true)
    .build();
```

- 您可以使用 ProfileCredentialsProvider.builder() 指定自定义配置文件的路径。

Example

```
ProfileCredentialsProvider profile = ProfileCredentialsProvider.builder()
    .profileFile(ProfileFile.builder().content(Paths.get("myProfileFile.file")).build())
    .build();
```

- 配置文件格式已更改，以更贴近于 AWS CLI。有关详细信息，请参阅《AWS Command Line Interface 用户指南》中的[配置 AWS CLI](#)。

版本 1.x 与 2.x 之间的凭证提供程序更改映射

AWSCredentialsProvider

更改类别	1.x	2.x
软件包/类名	com.amazonaws.auth .AWSCredentialsPro vider	software.amazon.aw ssdk.auth.credenti als.AwsCredentials Provider
方法名	getCredentials	resolveCredentials
不支持的方法	refresh	不支持

DefaultAWSCredentialsProviderChain

更改类别	1.x	2.x
软件包/类名	com.amazonaws.auth .DefaultAWSCredent ialsProviderChain	software.amazon.aw ssdk.auth.credenti als.DefaultCredent ialsProvider
创建	new DefaultAW SCredentialsProvid erChain	DefaultCredentials Provider.create

更改类别	1.x	2.x
不支持的方法	<code>getInstance</code>	不支持
外部设置的优先顺序	系统属性之前的环境变量	环境变量之前的系统属性

AWSStaticCredentialsProvider

更改类别	1.x	2.x
软件包/类名	<code>com.amazonaws.auth.AWSStaticCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.StaticCredentialsProvider</code>
创建	<code>new AWSStaticCredentialsProvider</code>	<code>StaticCredentialsProvider.create</code>

EnvironmentVariableCredentialsProvider

更改类别	1.x	2.x
软件包/类名	<code>com.amazonaws.auth.EnvironmentVariableCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider</code>
创建	<code>new EnvironmentVariableCredentialsProvider</code>	<code>EnvironmentVariableCredentialsProvider.create</code>
环境变量名	<code>AWS_ACCESS_KEY</code>	<code>AWS_ACCESS_KEY_ID</code>
	<code>AWS_SECRET_KEY</code>	<code>AWS_SECRET_ACCESS_KEY</code>

SystemPropertiesCredentialsProvider

更改类别	1.x	2.x
软件包/类名	<code>com.amazonaws.auth.SystemPropertiesCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.SystemPropertyCredentialsProvider</code>
创建	<code>new SystemPropertiesCredentialsProvider</code>	<code>SystemPropertiesCredentialsProvider.create</code>
系统属性名称	<code>aws.secretKey</code>	<code>aws.secretAccessKey</code>

ProfileCredentialsProvider

更改类别	1.x	2.x
软件包/类名	<code>com.amazonaws.auth.profile.ProfileCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider</code>
创建	<code>new ProfileCredentialsProvider</code>	<code>ProfileCredentialsProvider.create</code>
自定义配置文件的位置	<ul style="list-style-type: none"> <code>AWS_CREDENTIAL_PROFILES_FILE</code> 环境变量 <code>new ProfileCredentialsProvider</code> 	<ul style="list-style-type: none"> <code>AWS_SHARED_CREDENTIALS_FILE</code> 环境变量 <code>ProfileCredentialsProvider.builder</code>

ContainerCredentialsProvider

更改类别	1.x	2.x
软件包/类名	<code>com.amazonaws.auth.ContainerCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.ContainerCredentialsProvider</code>
创建	<code>new ContainerCredentialsProvider</code>	<code>ContainerCredentialsProvider.create</code>
指定异步刷新	不支持	默认行为

InstanceProfileCredentialsProvider

更改类别	1.x	2.x
软件包/类名	<code>com.amazonaws.auth.InstanceProfileCredentialsProvider</code>	<code>software.amazon.awssdk.auth.credentials.InstanceProfileCredentialsProvider</code>
创建	<code>new InstanceProfileCredentialsProvider</code>	<code>InstanceProfileCredentialsProvider.create</code>
指定异步刷新	<code>new InstanceProfileCredentialsProvider(true)</code>	<code>InstanceProfileCredentialProvider.builder().asyncCredentialUpdateEnabled(true).build()</code>
系统属性名称	<code>com.amazonaws.sdk.disableEc2Metadata</code>	<code>aws.disableEc2Metadata</code>

更改类别	1.x	2.x
	<code>com.amazonaws.sdk.ec2MetadataServiceEndpointOverride</code>	<code>aws.ec2MetadataServiceEndpoint</code>

STSAssumeRoleSessionCredentialsProvider

更改类别	1.x	2.x
软件包/类名	<code>com.amazonaws.auth.STSAssumeRoleSessionCredentialsProvider</code>	<code>software.amazon.awssdk.services.sts.auth.StsAssumeRoleCredentialsProvider</code>
创建	<ul style="list-style-type: none"> <code>new STSAssumeRoleSessionCredentialsProvider</code> <code>new STSAssumeRoleSessionCredentialsProvider.Builder</code> 	<code>StsAssumeRoleCredentialsProvider.builder</code>
异步刷新	默认行为	默认行为
配置	<code>new STSAssumeRoleSessionCredentialsProvider.Builder</code>	配置 <code>StsClient</code> 和 <code>AssumeRoleRequest</code> 请求

STSSessionCredentialsProvider

更改类别	1.x	2.x
软件包/类名	<code>com.amazonaws.auth.STSSessionCredentialsProvider</code>	<code>software.amazon.awssdk.services.sts.auth.StsGetSessionTokenCredentialsProvider</code>
创建	<code>new STSAssumeRoleSessionCredentialsProvider</code>	<code>StsGetSessionTokenCredentialsProvider.builder</code>
异步刷新	默认行为	<code>StsGetSessionTokenCredentialsProvider.builder</code>
配置	构造器参数	在生成器中配置 <code>an StsClient d GetSessionTokenRequest</code> 请求

WebIdentityFederationSessionCredentialsProvider

更改类别	1.x	2.x
软件包/类名	<code>com.amazonaws.auth.WebIdentityFederationSessionCredentialsProvider</code>	<code>software.amazon.awssdk.services.sts.auth.StsAssumeRoleWithWebIdentityCredentialsProvider</code>
创建	<code>new WebIdentityFederationSessionCredentialsProvider</code>	<code>StsAssumeRoleWithWebIdentityCredentialsProvider.builder</code>

更改类别	1.x	2.x
异步刷新	默认行为	<code>StsAssumeRoleWithWebIdentityCredentialsProvider.builder</code>
配置	构造器参数	在生成器中配置 <code>an StsClient d AssumeRoleWithWebIdentityRequest</code> 请求

类被替换

1.x 级	2.x 替换类
<code>com.amazonaws.auth.EC2ContainerCredentialsProviderWrapper</code>	<code>software.amazon.awssdk.auth.credentials.ContainerCredentialsProvider</code> 和 <code>software.amazon.awssdk.auth.credentials.InstanceProfileCredentialsProvider</code>
<code>com.amazonaws.services.s3.S3CredentialsProviderChain</code>	<code>software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider</code> 和 <code>software.amazon.awssdk.auth.credentials.AnonymousCredentialsProvider</code>

类已删除

1.x 级
<code>com.amazonaws.auth.ClasspathPropertiesFileCredentialsProvider</code>
<code>com.amazonaws.auth.PropertiesFileCredentialsProvider</code>

地区变化

本部分介绍了AWS SDK for Java 2.x 中针对 Region 和 Regions 类的使用实施的更改。

区域配置

- 一些 AWS 服务没有特定于区域的端点。在使用这些服务时，您必须将区域设置为 `Region.AWS_GLOBAL` 或 `Region.AWS_CN_GLOBAL`。

Example

```
Region region = Region.AWS_GLOBAL;
```

- `com.amazonaws.regions.Regions` 和 `com.amazonaws.regions.Region` 类现在合并成一个类 `software.amazon.awssdk.regions.Region`。

方法和类名映射

下表列出了AWS SDK for Java 的 1.x 与 2.x 版之间的区域相关类的映射。您可以使用 `of()` 方法创建这些类的实例。

Example

```
RegionMetadata regionMetadata = RegionMetadata.of(Region.US_EAST_1);
```

1.x Regions 类方法更改

1.x	2.x
<code>Regions.fromName</code>	<code>Region.of</code>
<code>Regions.getName</code>	<code>Region.id</code>
<code>Regions.getDescription</code>	<code>Region.metadata().description()</code>
<code>Regions.getCurrentRegion</code>	不支持
<code>Regions.DEFAULT_REGION</code>	不支持
<code>Regions.name</code>	<code>Region.id</code>

1.x 区域类方法更改

1.x	2.x
<code>Region.getName</code>	<code>Region.id</code>
<code>Region.hasHttpsEndpoint</code>	不支持
<code>Region.hasHttpEndpoint</code>	不支持
<code>Region.getAvailableEndpoints</code>	不支持
<code>Region.createClient</code>	不支持

RegionMetadata 类方法更改

1.x	2.x
<code>RegionMetadata.getName</code>	<code>RegionMetadata.name</code>
<code>RegionMetadata.getDomain</code>	<code>RegionMetadata.domain</code>
<code>RegionMetadata.getPartition</code>	<code>RegionMetadata.partition</code>

ServiceMetadata 类方法更改

1.x	2.x
<code>Region.getServiceEndpoint</code>	<code>ServiceMetadata.endpointFor(Region)</code>
<code>Region.isServiceSupported</code>	<code>ServiceMetadata.regions().contains(Region)</code>

操作、请求和响应变更

在 Java 版 SDK 的 v2.x 中，请求会传递给客户端操作。例如 `DynamoDbClient'sPutItemRequest`，传递给 `DynamoDbClient.putItem` 操作。这些操作会返回来自的响应 AWS 服务，例如 `PutItemResponse`。

适用于 Java 的 SDK 版本 2.x 与 1.x 相比有以下变化。

- 现在，具有多个响应页面的Paginator操作可以自动遍历响应中的所有项目。
- 您不能改变请求和响应。
- 必须使用静态生成器方法而不是构造函数来创建请求和响应。例如，现在new PutItemRequest().withTableName(...)PutItemRequest.builder().tableName(...).build() 1.x。
- 操作支持创建请求的简短方法:dynamoDbClient.putItem(request -> request.tableName(...)).

流式操作

诸如 Amazon S3 getObject 和putObject方法之类的流操作现在支持非阻塞 I/O。因此，请求和响应 POJO 不再将InputStream作为参数。相反，对于同步请求RequestBody，请求对象接受的是字节流。异步等效项接受AsyncRequestBody。

Example 1.x 中的 Amazon S3 putObject 操作

```
s3client.putObject(BUCKET, KEY, new File(file_path));
```

Example 2.x 中的 Amazon S3 putObject 操作

```
s3client.putObject(PutObjectRequest.builder()  
    .bucket(BUCKET)  
    .key(KEY)  
    .build(),  
    RequestBody.of(Paths.get("myfile.in")));
```

并行地，ResponseTransformer对于同步客户端，流式响应对象接受 a，AsyncResponseTransformer为异步客户端接受 a。

Example 1.x 中的 Amazon S3 getObject 操作

```
S3Object o = s3.getObject(bucket, key);  
S3ObjectInputStream s3is = o.getObjectContent();  
FileOutputStream fos = new FileOutputStream(new File(key));
```

Example 2.x 中的 Amazon S3 `getObject` 操作

```
s3client.getObject(GetObjectRequest.builder().bucket(bucket).key(key).build(),
    ResponseTransformer.toFile(Paths.get("key")));
```

在 Java SDK for Java 2.x 中，流式响应操作有一种 `AsBytes` 方法可以将响应加载到内存中并简化常见的内存类型转换。

Exception 更改

异常类的名称、它们的结构和它们的关系都发生了变化。

`software.amazon.awssdk.core.exception.SdkException` 是所有其他异常都扩展的新基 `Exception` 类。

此表列出了 `Exception` 类名更改的映射。

1.x	2.x
<code>com.amazonaws.SdkBaseException</code> <code>com.amazonaws.AmazonClientException</code>	<code>software.amazon.awssdk.core.exception.SdkException</code>
<code>com.amazonaws.SdkClientException</code>	<code>software.amazon.awssdk.core.exception.SdkClientException</code>
<code>com.amazonaws.AmazonServiceException</code>	<code>software.amazon.awssdk.awscore.exception.AwsServiceException</code>

下表列出了版本 1.x 与 2.x 之间 `Exception` 类方法的映射。

1.x	2.x
<code>AmazonServiceException.getRequestId</code>	<code>SdkServiceException.requestId</code>
<code>AmazonServiceException.getServiceName</code>	<code>AwsServiceException.awsErrorDetails().serviceName</code>

1.x	2.x
<code>AmazonServiceException.getErrorCode</code>	<code>AwsServiceException.awsErrorDetails().errorCode</code>
<code>AmazonServiceException.getErrorMessage</code>	<code>AwsServiceException.awsErrorDetails().errorMessage</code>
<code>AmazonServiceException.getStatusCode</code>	<code>AwsServiceException.awsErrorDetails().sdkHttpResponse().statusCode</code>
<code>AmazonServiceException.getHttpHeaders</code>	<code>AwsServiceException.awsErrorDetails().sdkHttpResponse().headers</code>
<code>AmazonServiceException.getRawResponse</code>	<code>AwsServiceException.awsErrorDetails().rawResponse</code>

序列化更改

SDK for Java v1.x 和 v2.x 的不同之处在于它们如何序列化列表对象以请求参数。

SDK for Java 1.x 不会序列化空列表，而 SDK for Java 2.x 会将空列表序列化为空参数。

例如，假设一项服务具有 `SampleOperation`，会接收 `SampleRequest`。`SampleRequest` 接受两个参数，字符串类型 `str1` 和列表类型 `listParam`，如以下示例所示。

Example 1.x 中 **SampleOperation** 的示例

```
SampleRequest v1Request = new SampleRequest()
    .withStr1("TestName");

sampleServiceV1Client.sampleOperation(v1Request);
```

线级日志记录显示 `listParam` 参数未序列化。

```
Action=SampleOperation&Version=2011-01-01&str1=TestName
```

Example 2.x 中 **SampleOperation** 的示例

```
sampleServiceV2Client.sampleOperation(b -> b
    .str1("TestName"));
```

线程级日志记录显示 `listParam` 参数进行了序列化，没有任何值。

```
Action=SampleOperation&Version=2011-01-01&str1=TestName&listParam=
```

特定于服务的更改

亚马逊 S3 的变化

默认情况下，适用于 Java 的 SDK 2.x 禁用匿名访问。因此，您必须使用启用匿名访问 `AnonymousCredentialsProvider`。

操作名称更改

Amazon S3 客户端的许多操作名称在 AWS SDK for Java 2.x 中已更改。在版本 1.x 中，不直接从服务 API 生成 Amazon S3 客户端。这会导致开发工具包操作与服务 API 之间的不一致。在版本 2.x 中，我们现在生成 Amazon S3 客户端以提高与服务 API 的一致性。

下表显示了两个版本中的操作名称。

亚马逊 S3 操作名称

1.x	2.x
<code>abortMultipartUpload</code>	<code>abortMultipartUpload</code>
<code>changeObjectStorageClass</code>	<code>copyObject</code>
<code>completeMultipartUpload</code>	<code>completeMultipartUpload</code>
<code>copyObject</code>	<code>copyObject</code>
<code>copyPart</code>	<code>uploadPartCopy</code>
<code>createBucket</code>	<code>createBucket</code>
<code>deleteBucket</code>	<code>deleteBucket</code>

1.x	2.x
<code>deleteBucketAnalyticsConfiguration</code>	<code>deleteBucketAnalyticsConfiguration</code>
<code>deleteBucketCrossOriginConfiguration</code>	<code>deleteBucketCors</code>
<code>deleteBucketEncryption</code>	<code>deleteBucketEncryption</code>
<code>deleteBucketInventoryConfiguration</code>	<code>deleteBucketInventoryConfiguration</code>
<code>deleteBucketLifecycleConfiguration</code>	<code>deleteBucketLifecycle</code>
<code>deleteBucketMetricsConfiguration</code>	<code>deleteBucketMetricsConfiguration</code>
<code>deleteBucketPolicy</code>	<code>deleteBucketPolicy</code>
<code>deleteBucketReplicationConfiguration</code>	<code>deleteBucketReplication</code>
<code>deleteBucketTaggingConfiguration</code>	<code>deleteBucketTagging</code>
<code>deleteBucketWebsiteConfiguration</code>	<code>deleteBucketWebsite</code>
<code>deleteObject</code>	<code>deleteObject</code>
<code>deleteObjectTagging</code>	<code>deleteObjectTagging</code>
<code>deleteObjects</code>	<code>deleteObjects</code>
<code>deleteVersion</code>	<code>deleteObject</code>
<code>disableRequesterPays</code>	<code>putBucketRequestPayment</code>
<code>doesBucketExist</code>	<code>headBucket</code>
<code>doesBucketExistV2</code>	<code>headBucket</code>

1.x	2.x
doesObjectExist	headObject
enableRequesterPays	putBucketRequestPayment
generatePresignedUrl	S3Presigner
getBucketAccelerateConfiguration	getBucketAccelerateConfiguration
getBucketAcl	getBucketAcl
getBucketAnalyticsConfiguration	getBucketAnalyticsConfiguration
getBucketCrossOriginConfiguration	getBucketCors
getBucketEncryption	getBucketEncryption
getBucketInventoryConfiguration	getBucketInventoryConfiguration
getBucketLifecycleConfiguration	getBucketLifecycle 或 getBucketLifecycleConfiguration
getBucketLocation	getBucketLocation
getBucketLoggingConfiguration	getBucketLogging
getBucketMetricsConfiguration	getBucketMetricsConfiguration
getBucketNotificationConfiguration	getBucketNotification 或 getBucketNotificationConfiguration
getBucketPolicy	getBucketPolicy
getBucketReplicationConfiguration	getBucketReplication
getBucketTaggingConfiguration	getBucketTagging
getBucketVersioningConfiguration	getBucketVersioning

1.x	2.x
<code>getBucketWebsiteConfiguration</code>	<code>getBucketWebsite</code>
<code>getObject</code>	<code>getObject</code>
<code>getObjectAcl</code>	<code>getObjectAcl</code>
<code>getObjectAsString</code>	<code>getObjectAsBytes().asUtf8String</code>
<code>getObjectMetadata</code>	<code>headObject</code>
<code>getObjectTagging</code>	<code>getObjectTagging</code>
<code>getResourceUrl</code>	S3Utilities#getUrl
<code>getS3AccountOwner</code>	<code>listBuckets</code>
<code>getUrl</code>	S3Utilities#getUrl
<code>headBucket</code>	<code>headBucket</code>
<code>initiateMultipartUpload</code>	<code>createMultipartUpload</code>
<code>isRequesterPaysEnabled</code>	<code>getBucketRequestPayment</code>
<code>listBucketAnalyticsConfigurations</code>	<code>listBucketAnalyticsConfigurations</code>
<code>listBucketInventoryConfigurations</code>	<code>listBucketInventoryConfigurations</code>
<code>listBucketMetricsConfigurations</code>	<code>listBucketMetricsConfigurations</code>
<code>listBuckets</code>	<code>listBuckets</code>
<code>listMultipartUploads</code>	<code>listMultipartUploads</code>
<code>listNextBatchOfObjects</code>	<code>listObjectsV2Paginator</code>
<code>listNextBatchOfVersions</code>	<code>listObjectVersionsPaginator</code>

1.x	2.x
<code>listObjects</code>	<code>listObjects</code>
<code>listObjectsV2</code>	<code>listObjectsV2</code>
<code>listParts</code>	<code>listParts</code>
<code>listVersions</code>	<code>listObjectVersions</code>
<code>putObject</code>	<code>putObject</code>
<code>restoreObject</code>	<code>restoreObject</code>
<code>restoreObjectV2</code>	<code>restoreObject</code>
<code>selectObjectContent</code>	<code>selectObjectContent</code>
<code>setBucketAccelerateConfiguration</code>	<code>putBucketAccelerateConfiguration</code>
<code>setBucketAcl</code>	<code>putBucketAcl</code>
<code>setBucketAnalyticsConfiguration</code>	<code>putBucketAnalyticsConfiguration</code>
<code>setBucketCrossOriginConfiguration</code>	<code>putBucketCors</code>
<code>setBucketEncryption</code>	<code>putBucketEncryption</code>
<code>setBucketInventoryConfiguration</code>	<code>putBucketInventoryConfiguration</code>
<code>setBucketLifecycleConfiguration</code>	<code>putBucketLifecycle</code> 或 <code>putBucketLifecycleConfiguration</code>
<code>setBucketLoggingConfiguration</code>	<code>putBucketLogging</code>
<code>setBucketMetricsConfiguration</code>	<code>putBucketMetricsConfiguration</code>
<code>setBucketNotificationConfiguration</code>	<code>putBucketNotification</code> 或 <code>putBucketNotificationConfiguration</code>

1.x	2.x
setBucketPolicy	putBucketPolicy
setBucketReplicationConfiguration	putBucketReplication
setBucketTaggingConfiguration	putBucketTagging
setBucketVersioningConfiguration	putBucketVersioning
setBucketWebsiteConfiguration	putBucketWebsite
setObjectAcl	putObjectAcl
setObjectRedirectLocation	copyObject
setObjectTagging	putObjectTagging
uploadPart	uploadPart

亚马逊 SNS 发生了变化

除配置为访问的区域外，SNS 客户端无法再访问其他区域中的 SNS 主题。

亚马逊 SQS 变更

SQS 客户端无法再访问其配置为访问的区域以外的区域中的 SQS 队列。

亚马逊 RDS 变更

适用于 Java 2.x 的 SDK 代替 1.x RdsIamAuthTokenGenerator 中的类。RdsUtilities#generateAuthenticationToken

配置文件更改

AWS SDK for Java 2.x 解析中的配置文件定义 `~/.aws/config`，`~/.aws/credentials` 以更紧密地模拟 CL AWS I 解析文件的方式。

适用于 Java 的 SDK 2.x：

- 通过按~顺序选中、(仅限 Windows)、(仅限 Windows)、\$USERPROFILE (仅限 Windows) \$HOME，然后选中user.home系统属性\$HOMEDRIVE，\$HOMEPATH在路径开头解析文件系统的默认路径分隔符。~/
- 查找AWS_SHARED_CREDENTIALS_FILE环境变量而不是AWS_CREDENTIAL_PROFILES_FILE。
- 以静默方式删除配置文件中配置文件名称开头不带单词profile的配置文件定义。
- 静默删除不包含字母数字、下划线或破折号字符的配置文件定义 (在配置文件中删除了前导profile单词之后)。
- 合并同一文件中重复的配置文件定义设置。
- 合并配置文件和凭据文件中重复的配置文件定义设置。
- 如果两个[profile foo]和位于同一个文件中[foo]，则不合并设置。
- [profile foo]如果在配置文件中同时找到[profile foo]和中的设置，[foo]则使用中的设置。
- 使用同一文件和配置文件中最后一次复制的设置的值。
- 可同时识别;和#用于定义注释。
- 识别;并在配置文件定义#中定义注释，即使字符与右括号相邻。
- 只有在设置值前面有空格时，才能识别;和#定义注释。
- 如果设置值前面没有空格，则可以识别;#和之后的所有内容。
- 将基于角色的证书视为优先级最高的证书。如果用户指定了属性，2.x SDK 将始终使用基于角色的凭证。role_arn
- 将基于会话的凭证视为凭证。second-highest-priority 如果未使用基于角色的凭证且用户指定了和属性，则 2.x SDK 将始终使用基于会话的凭证。aws_access_key_id aws_session_token
- 如果未使用基于角色和基于会话的凭证并且用户指定了属性，则使用基本凭证。aws_access_key_id

环境变量和系统属性发生变化

1.x 环境变量	1.x 系统属性	2.x 环境变量	2.x 系统属性
AWS_ACCESS_KEY_ID AWS_ACCESS_KEY	aws.accessKeyId	AWS_ACCESS_KEY_ID	aws.accessKeyId

1.x 环境变量	1.x 系统属性	2.x 环境变量	2.x 系统属性
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_CREDENTIALS_PROFILE_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.ec2MetadataServiceEndpointOverride	AWS_EC2_METADATA_SERVICE_ENDPOINT	aws.ec2MetadataServiceEndpoint
AWS_CONTAINER_CREDENTIALS_RELATIVE_URI		AWS_CONTAINER_CREDENTIALS_RELATIVE_URI	aws.containerCredentialsPath

1.x 环境变量	1.x 系统属性	2.x 环境变量	2.x 系统属性
AWS_CONTAINER_CREDENTIALS_FULL_URI		AWS_CONTAINER_CREDENTIALS_FULL_URI	aws.containerCredentialsFullUri
AWS_CONTAINER_AUTHORIZATION_TOKEN		AWS_CONTAINER_AUTHORIZATION_TOKEN	aws.containerAuthorizationToken
AWS_CBOR_DISABLED	com.amazonaws.sdk.disableCbor	CBOR_ENABLED	aws.cborEnabled
AWS_ION_BINARY_DISABLE	com.amazonaws.sdk.disableIonBinary	BINARY_ION_ENABLED	aws.binaryIonEnabled
AWS_EXECUTION_ENV		AWS_EXECUTION_ENV	aws.executionEnvironment
	com.amazonaws.sdk.disableCertChecking	不支持 (请求功能)	不支持 (请求功能)
	com.amazonaws.sdk.enableDefaultMetrics	不支持	不支持

1.x 环境变量	1.x 系统属性	2.x 环境变量	2.x 系统属性
	<code>com.amazonaws.sdk.enableThrottledRetry</code>	不支持	不支持
	<code>com.amazonaws.regions.RegionUtils.fileOverride</code>	不支持 (请求功能)	不支持 (请求功能)
	<code>com.amazonaws.regions.RegionUtils.disableRemote</code>	不支持 (请求功能)	不支持 (请求功能)
	<code>com.amazonaws.services.s3.disableImplicitGlobalClients</code>	不支持 (请求功能)	不支持 (请求功能)
	<code>com.amazonaws.sdk.enableInRegionOptimizedMode</code>	不支持 (请求功能)	不支持 (请求功能)

服务员从版本 1 到版本 2 的变化

本主题详细介绍了 Waiters 功能从版本 1 (v1) 到版本 2 (v2) 的变化。

下表具体说明了 DynamoDB 服务员的区别。其他服务的服务员也遵循同样的模式。

高级别更改

服务员的职业与该服务位于同一个 Maven 工件中。

更改	v1	v2
Maven 依赖项	<pre><dependencyManagement> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> 1.12.680¹</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> </dependencies></pre>	<pre><dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> 2.25.10²</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></pre>
软件包名称	com.amazonaws.serv ices.dynamodbv2.wa iters	software.amazon.aw ssdk.services.dyna modb.waiters

更改	v1	v2
类名	AmazonDynamoDBWaiters	<ul style="list-style-type: none"> • 同步：DynamoDbWaiter • 异步：DynamoDbAsyncWaiter

¹ [最新版本](#)。 ² [最新版本](#)。

API 变更

更改	v1	v2
创建服务员	<pre>AmazonDynamoDB client = AmazonDynamoDBClientBuilder .standard().build(); AmazonDynamoDBWaiters waiter = client.waiters();</pre>	<p>同步：</p> <pre>DynamoDbClient client = DynamoDbClient.create(); DynamoDbWaiter waiter = client.waiter();</pre> <p>异步：</p> <pre>DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create(); DynamoDbAsyncWaiter waiter = asyncClient.waiter();</pre>
等到表存在	<p>同步：</p> <pre>waiter.tableExists() .run(new WaiterParameters<>(new DescribeTableRequest(tableName)));</pre>	<p>同步：</p> <pre>WaiterResponse<DescribeTableResponse> waiterResponse = waiter.waitUntilTableExists(r -> r.tableName("myTable"));</pre>

更改	v1	v2
	<p>异步：</p> <pre> waiter.tableExists() .runAsync(new WaiterParameters() .withRequest(new DescribeTableReque st(tableName)), new WaiterHan dler() { @Override public void onWaitSuccess(AmazonWebServiceRe quest amazonWeb ServiceRequest) { System.out.println ("Table creation succeeded"); } @Override public void onWaitFai lure(Exception e) { e.printStackTrace(); } }).get(); </pre>	<pre> waiterResponse.match ed().response() .ifPresen t(System.out::prin tln); </pre> <p>异步：</p> <pre> waiter.waitUntilTa bleExists(r -> r.tableName(tableName)) .whenComp lete((r, t) -> { if (t != null) { t.printStackTrace(); } else { System.out.println("Table creation succeeded"); } }).join(); </pre>

配置更改

更改	v1	v2
轮询策略 (最大尝试次数和固定延迟)	<pre> 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); </pre>	<pre> FixedDelayBackoffStrategy fixedDelayBackoffStrategy = FixedDelayBackoffStrategy.create(Duration.ofSeconds(3)); waiter.waitUntilTableExists(r -> r.tableName(tableName), c -> c.maxAttempts(10) .backoffStrategy(fixedDelayBackoffStrategy)); </pre>

Amazon S3 Transfer Manager 版本 1 到 版本 2 的更改

本主题详细介绍了 Amazon S3 Transfer Manager 从版本 1 (v1) 到版本 2 (v2) 的更改。

高级别更改

更改	v1	v2
Maven 依赖项	<pre> <dependencyManagement> <dependencies> <dependency> <groupId> com.amazonaws</gro groupId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> 1.12.587¹</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManagem ent> <dependencies> <dependency> <groupId> com.amazonaws</gro groupId> <artifact Id>aws-java-sdk-s3</ artifactId> </dependency> </dependencies> </pre>	<pre> <dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> 2.21.21²</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManagem ent> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifactId>s3- transfer-manager</art ifactId> </dependency> <dependency> <groupId> software.amazon.aw ssdk.crt</groupId> <artifact Id>aws-crt</artifa ctId> <version> 0.28.7³</version> </dependency> </dependencies> </pre>

更改	v1	v2
软件包名称	com.amazonaws.serv ices.s3.transfer	software.amazon.aw ssdk.transfer.s3
类名称	TransferManager	S3TransferManager

¹ [最新版本](#)。 ² [最新版本](#)。 ³ [最新版本](#)。

配置 API 更改

设置	v1	v2
(获取生成器)	<pre>TransferManagerBuilder tmBuilder = TransferManagerBui lder.standard();</pre>	<pre>S3TransferManager. Builder tmBuilder = S3TransferManager. builder();</pre>
S3 客户端	<pre>tmBuilder.withS3Cl ient(...); tmBuilder.setS3C lient(...);</pre>	<pre>tmBuilder.s3Client (...);</pre>
执行程序	<pre>tmBuilder.withExec utorFactory(...); tmBuilder.setExecu torFactory(...);</pre>	<pre>tmBuilder.executor (...);</pre>
关闭线程池	<pre>tmBuilder.withShut DownThreadPools(...); tmBuilder.setS hutdownThreadPools (...);</pre>	不支持。S3 TransferManager 关闭后，提供的执行器不会关闭
最小上传段大小	<pre>tmBuilder.withMini mumUploadPartSize(...);</pre>	<pre>S3AsyncClient s3 = S3AsyncClient.crtB uilder().</pre>

设置	v1	v2
	<pre>tmBuilder.setMinimumUploadPartSize(...);</pre>	<pre>minimumPartSizeInBytes(...) .build(); tmBuilder.s3Client(s3);</pre>
分段上传阈值	<pre>tmBuilder.withMinimumUploadPartSize(...); tmBuilder.setMinimumUploadPartSize(...);</pre>	<pre>S3AsyncClient s3 = S3AsyncClient.crtBuilder() .thresholdInBytes(...) .build(); tmBuilder.s3Client(s3);</pre>
最小复制段大小	<pre>tmBuilder.withMinimumUploadPartSize(...); tmBuilder.setMinimumUploadPartSize(...);</pre>	<pre>S3AsyncClient s3 = S3AsyncClient.crtBuilder() .minimumPartSizeInBytes(...) .build(); tmBuilder.s3Client(s3);</pre>
分段复制阈值	<pre>tmBuilder.withMinimumUploadPartSize(...); tmBuilder.setMinimumUploadPartSize(...);</pre>	<pre>S3AsyncClient s3 = S3AsyncClient.crtBuilder() .thresholdInBytes(...) .build(); tmBuilder.s3Client(s3);</pre>

设置	v1	v2
禁用并行下载	<pre>tmBuilder.withDisableParallelDownloads(...); tmBuilder.setDisableParallelDownloads(...);</pre>	<p>通过将基于 Java 的标准 S3 客户端传递给 Transfer Manager 来禁用并行下载。</p> <pre>S3AsyncClient s3 = S3AsyncClient.builder().build(); tmBuilder.s3Client(s3);</pre>
始终计算多分段 md5	<pre>tmBuilder.withAlwaysCalculateMultipartMd5(...); tmBuilder.setAlwaysCalculateMultipartMd5(...);</pre>	不支持。

行为更改

并行传输需要 AWS 基于 CRT 的 S3 客户端

在 SDK for Java 2.x 中，自动并行传输功能（分段上传/下载）通过[基于 AWS CRT 的 S3 客户端](#)提供。要启用并行传输功能，必须明确添加[AWS 通用运行时系统 \(CRT\) 库](#)依赖项，以实现最高性能。

仅 AWS 基于 CRT 的 S3 客户端（不使用 S3TransferManager）即可最大限度地提高并行传输性能。S3TransferManagerv2 提供了其他 API，可以更轻松地传输文件和目录。

能否执行 S3TransferManager 并行传输取决于启动方式 S3TransferManager 以及 AWS 公共运行时 (CRT) 库是否已声明为依赖项。

下表描述了将 CRT 声明为依赖项和不将 AWS CRT 声明为依赖项的 S3TransferManager v2 的三种初始化方案。

S3 TransferManager v2 初始化方法	AWS CRT 是否被声明为依赖项？	
	是	否
<p>在不传递 S3AsyncClient 实例的情况下初始化 S3TransferManager</p> <p>静态创建方法：</p> <pre data-bbox="115 579 1019 657">S3TransferManager.create();</pre> <p>- 或者 -</p> <p>生成器方法：</p> <pre data-bbox="115 846 1019 924">S3TransferManager.builder().build();</pre>	 自动并行传输启用	 自动并行传输禁用
<p>传递一个使用 crt*() 生成器方法之一构建的 S3AsyncClient 实例</p> <pre data-bbox="115 1087 1019 1283">S3AsyncClient s3AsyncClient = S3AsyncClient.crtBuilder().build(); S3TransferManager.builder().s3AsyncClient(s3AsyncClient).build();</pre> <p>- 或者 -</p> <pre data-bbox="115 1394 1019 1589">S3AsyncClient s3AsyncClient = S3AsyncClient.crtCreate(); S3TransferManager.builder().s3AsyncClient(s3AsyncClient).build();</pre>	 自动并行传输启用	 运行时错误
<p>传递一个使用标准生成器方法之一构建的 S3AsyncClient 实例，以便 Transfer Manager 不引用 CRT</p> <pre data-bbox="115 1755 1019 1833">S3AsyncClient s3AsyncClient = S3AsyncClient.builder().build();</pre>	 自动并行传输禁用	 自动并行传输禁用

S3 TransferManager v2 初始化方法	AWS CRT 是否被声明为依赖项？	
<pre>S3TransferManager.builder().s3AsyncClient(s3AsyncClient).build();</pre> <p>- 或者 -</p> <pre>S3AsyncClient s3AsyncClient = S3AsyncClient.create(); S3TransferManager.builder().s3AsyncClient(s3AsyncClient).build();</pre>		

通过字节范围提取进行并行下载

启用自动并行传输功能后，S3 Transfer Manager v2 使用[字节范围提取](#)来并行检索对象的特定部分（分段下载）。使用 v2 下载对象的方式不取决于对象最初的上传方式。所有下载都可以从高吞吐量和并发性中受益。

相比之下，在 S3 Transfer Manager v1 中，对象最初的上传方式确实很重要。S3 Transfer Manager v1 检索对象各个分段的方式与上传分段的方式相同。如果对象最初是作为单个对象上传的，则 S3 Transfer Manager v1 无法通过使用子请求来加速下载过程。

失败行为

在 S3 Transfer Manager v1 中，如果任何子请求失败，则目录传输请求将失败。与 v1 不同，即使某些子请求失败，从 S3 Transfer Manager v2 返回的 future 也会成功完成。

因此，即使 future 成功完成，也应使用 [CompletedDirectoryDownload.failedTransfers\(\)](#) 方法或 [CompletedDirectoryUpload.failedTransfers\(\)](#) 方法检查响应中是否存在错误。

EC2 元数据实用程序从版本 1 到版本 2 的变化

本主题详细介绍了 SDK for Java Amazon Elastic Compute Cloud (EC2) 元数据实用程序从版本 1 (v1) 到版本 2 (v2) 的变化。

高级别更改

更改	v1	v2
Maven 依赖项	<pre> <dependencyManagement> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> 1.12.587¹</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> </pre>	<pre> <dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> 2.21.21²</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-client³</ artifactId> </dependency> </dependencies> </pre>

更改	v1	v2
软件包名称	com.amazonaws.util	software.amazon.awssdk.imds
实例化方法	<p>使用静态实用程序方法；不进行实例化：</p> <pre>String localHostName = EC2MetadataUtils.getLocalHostName();</pre>	<p>使用静态工厂方法：</p> <pre>Ec2MetadataClient client = Ec2MetadataClient.create();</pre> <p>或者使用生成器方法：</p> <pre>Ec2MetadataClient client = Ec2MetadataClient.builder() .endpointMode(EndpointMode.IPV6) .build();</pre>
客户端类型	仅同步实用程序方法： EC2MetadataUtils	同步：Ec2MetadataClient 异步：Ec2MetadataAsyncClient

¹ [最新版本](#)。² [最新版本](#)。

³ 注意 v2 的 `apache-client` 模块声明。EC2 元数据实用程序的 V2 需要为同步元数据客户端实现 `SdkHttpClient` 接口，或为异步元数据客户端实现 `SdkAsyncHttpClient` 接口。[???](#)部分显示了您可以使用的 HTTP 客户端列表。

请求元数据

在 v1 中，您可以使用不接受任何参数的静态方法来请求 EC2 资源的元数据。相比之下，在 v2 中，您需要将 EC2 资源的路径指定为参数。下表显示了不同的方法。

v1	v2
<pre>String userMetaData = EC2Metada taUtils.getUserData();</pre>	<pre>Ec2MetadadataClient client = Ec2Metada taClient.create(); Ec2MetadadataResponse response = client.get("/latest/ user-data"); String userMetaData = response.asString();</pre>

请参阅[实例元数据类别](#)，查找请求元数据时需要提供的路径。

Note

在 v2 中使用实例元数据客户端时，您应该针对检索元数据的所有请求使用同一客户端。

行为更改

JSON 数据

在 EC2 上，本地运行的实例元数据服务 (IMDS) 以 JSON 格式的字符串形式返回一些元数据。[实例身份文档](#)的动态元数据就是一个例子。

v1 API 针对每种实例身份元数据包含不同的方法，而 v2 API 会直接返回 JSON 字符串。要处理 JSON 字符串，您可以使用[文档 API](#) 解析响应并浏览 JSON 结构。

下表比较了在 v1 和 v2 中检索实例身份文档元数据的方式。

应用场景	v1	v2
检索区域	<pre>InstanceInfo instanceI nfo = EC2Metada taUtils.getInstanc eInfo();</pre>	<pre>Ec2MetadadataResponse response = client.get("/lates t/dynamic/instance- identity/document");</pre>

应用场景	v1	v2
	<pre>String region = instanceInfo.getRegion();</pre>	<pre>Document instanceInfo = response.asDocument(); String region = instanceInfo.asMap() .get("region").asString();</pre>
检索实例 ID	<pre>InstanceInfo instanceInfo = EC2MetadataUtils.getInstanceInfo(); String instanceId = instanceInfo.getInstanceId();</pre>	<pre>Ec2MetadataResponse response = client.get("/latest/dynamic/instance-identity/document"); Document instanceInfo = response.asDocument(); String instanceId = instanceInfo.asMap() .get("instanceId").asString();</pre>
检索实例类型	<pre>InstanceInfo instanceInfo = EC2MetadataUtils.getInstanceInfo(); String instanceType = instanceInfo.getInstanceType();</pre>	<pre>Ec2MetadataResponse response = client.get("/latest/dynamic/instance-identity/document"); Document instanceInfo = response.asDocument(); String instanceType = instanceInfo.asMap() .get("instanceType").asString();</pre>

端点解析差异

下表显示了 SDK 为将端点解析到 IMDS 而检查的位置。这些位置按优先级降序列出。

v1	v2
系统属性： <code>com.amazonaws.sdk.ec2MetadataServiceEndpointOverride</code>	客户端生成器配置方法： <code>endpoint(...)</code>
环境变量： <code>AWS_EC2_METADATA_SERVICE_ENDPOINT</code>	系统属性： <code>aws.ec2MetadataServiceEndpoint</code>
默认值： <code>http://169.254.169.254</code>	Config 文件： <code>~.aws/config</code> ，及 <code>ec2_metadata_service_endpoint</code> 设置
	与已解析的 <code>endpoint-mode</code> 相关联的值
	默认值： <code>http://169.254.169.254</code>

v2 中的端点解析

如果您使用生成器显式设置端点，该端点值的优先级高于所有其他设置。当以下代码执行时，`aws.ec2MetadataServiceEndpoint` 系统属性和 config 文件 `ec2_metadata_service_endpoint` 设置如果存在，将被忽略。

```
Ec2MetadataClient client = Ec2MetadataClient
    .builder()
    .endpoint(URI.create("endpoint.to.use"))
    .build();
```

端点模式

在 v2 中，您可以指定端点模式，将元数据客户端配置为针对 IPv4 或 IPv6 使用默认端点值。端点模式不适用于 v1。用于 IPv4 的默认值为 `http://169.254.169.254`，对于 IPv6，则为 `http://[fd00:ec2::254]`。

下表按优先级降序显示了设置端点模式时可以采用的不同方法。

		可能的值
客户端生成器配置方法： <code>endpointMode(...)</code>	<pre>Ec2MetadataClient client = Ec2Metada taClient .builder() .endpointMode(Endp ointMode.IPV4) .build();</pre>	EndpointMode.IPV4 , EndpointMode.IPV6
系统属性	<code>aws.ec2MetadataServiceEndpointMode</code>	IPv4、IPv6 (大小写没有影响)
Config 文件： <code>~/.aws/config</code>	<code>ec2_metadata_service_endpoint</code> 设置	IPv4、IPv6 (大小写没有影响)
未在前面的方法中指定	使用 IPv4	

在 v2 中，SDK 如何解析 **endpoint** 或 **endpoint-mode**

1. SDK 使用您通过客户端生成器在代码中设置的值，并忽略任何外部设置。由于在客户端生成器上同时调用 `endpoint` 和 `endpointMode` 时，SDK 会抛出异常，因此 SDK 将使用您所用的任一方法中的端点值。
2. 如果您没有在代码中设置值，SDK 会查看外部配置；首先查找系统属性，然后是 config 文件中的设置。
 - a. SDK 会先检查端点值。如果找到值，则使用该值。
 - b. 如果 SDK 仍未找到值，则会查找端点模式设置。
3. 最后，如果 SDK 未找到任何外部设置，并且您未在代码中配置元数据客户端，则 SDK 会使用 IPv4 值 `http://169.254.169.254`。

IMDSv2

Amazon EC2 定义了两种访问实例元数据的方法：

- 实例元数据服务版本 1 (IMDSv1)：请求/响应方法
- 实例元数据服务版本 2 (IMDSv2)：面向会话的方法

下表比较了 Java SDK 与 IMDS 协同工作的方式。

v1	v2
默认使用 IMDSv2	始终使用 IMDSv2
尝试为每个请求获取会话令牌，如果未能获取会话令牌，则回退到 IMDSv1	将会话令牌保存在内部缓存中，该令牌可重复用于多个请求

SDK for Java 2.x 仅支持 IMDSv2，不会回退到 IMDSv1。

配置差异

下表列出了不同的配置选项。

配置	v1	v2
重试	配置不可用	可通过生成器方法 <code>retryPolicy(...)</code> 配置
HTTP	连接超时可通过 <code>AWS_METADATA_SERVICE_TIMEOUT</code> 环境变量配置。默认值为 1 秒。	通过将 HTTP 客户端传递给生成器方法 <code>httpClient(...)</code> 即可进行配置。HTTP 客户端的默认连接超时为 2 秒。

v2 HTTP 配置示例

以下示例演示了如何配置元数据客户端。此示例配置连接超时并使用 Apache HTTP 客户端。

```

SdkHttpClient httpClient = ApacheHttpClient.builder()
    .connectionTimeout(Duration.ofSeconds(1))
    .build();

Ec2MetadataClient imdsClient = Ec2MetadataClient.builder()
    .httpClient(httpClient)
    .build();

```

Amazon CloudFront 预签名从版本 1 更改为版本 2

本主题详细介绍了 Amazon CloudFront 从版本 1 (v1) 到版本 2 (v2) 的变化。

高级别更改

更改	v1	v2
Maven 依赖项	<pre><dependencyManagement> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> 1.12.587¹</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>cloudfront</art ifactId> </dependency> </dependencies> </dependencies></pre>	<pre><dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> 2.21.21²</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>cloudfront</art ifactId> </dependency> </dependencies></pre>
软件包名称	com.amazonaws.serv ices.cloudfront	software.amazon.aw ssdk.services.clou dfront

更改	v1	v2
类名	CloudFrontUrlSigner CloudFrontCookieSigner	CloudFrontUtilities SignedUrl CannedSignerRequest CustomSignerRequest

¹ [最新版本](#)。 ² [最新版本](#)。

API 变更

行为	v1	v2
创建预设请求	参数直接传递给 API。	<pre>CannedSignerRequest cannedRequest = CannedSig nerRequest.builder() .resourceUrl(resou rceUrl) .privateKey(privat eKey) .keyPairId(keyPairId) .expirationDate(ex pirationDate) .build();</pre>
生成自定义请求	参数直接传递给 API。	<pre>CustomSignerRequest customRequest = CustomSig nerRequest.builder()</pre>

行为	v1	v2
		<pre> .resourceUrl(resourceUrl) .privateKey(keyFile) .keyPairId(keyPairId) .expirationDate(expirationDate) .activeDate(activeDate) .ipRange(ipRange) .build(); </pre>
生成签名 URL (固定)	<pre> String signedUrl = CloudFrontUrlSigner.getSignedURLWithCannedPolicy(resourceUrl, keyPairId, privateKey, expirationDate); </pre>	<pre> CloudFrontUtilities cloudFrontUtilities = CloudFrontUtilities.create(); SignedUrl signedUrl = cloudFrontUtilities.getSignedUrlWithCannedPolicy(cannedRequest); String url = signedUrl.url(); </pre>

行为	v1	v2
生成签名的 cookie (自定义)	<pre> CookiesForCustomPolicy cookies = CloudFrontCookieSi gner.getCookiesFor CustomPolicy(resourceUrl, privateKey, keyPairId , expirationDate, activeDate, ipRange); </pre>	<pre> CloudFrontUtilities cloudFrontUtilities = CloudFrontUtilitie s.create(); CookiesForCustomPolicy cookies = cloudFrontUtilitie s.getCookiesForCus tomPolicy(customRe quest); </pre>

v2 中重构的 cookie 标头

在 Java v1 中，Java 软件开发工具包将 cookie 标头作为 `Map.Entry<String, String>` 提供。

```

Map.Entry<String, String> signatureMap = cookies.getSignature();
String signatureKey = signatureMap.getKey(); // "CloudFront-Signature"
String signatureValue = signatureMap.getValue(); // "[SIGNATURE_VALUE]"

```

Java v2 SDK 将整个标头作为一个单独 `String` 的标题提供。

```

String signatureHeaderValue = cookies.signatureHeaderValue(); // "CloudFront-
Signature=[SIGNATURE_VALUE]"

```

解析从版本 1 到版本 2 的 Amazon S3 URI 时发生的变化

本主题详细介绍了在解析从版本 1 (v1) 到版本 2 (v2.) 的 Amazon S3 URI 时所做的更改。

高级别更改

要在 v1 中开始解析 S3 URI，请使用构造函数实例 `AmazonS3URI` 化。在 v2 中，你调用 `parseUri()` 一个的实例 `S3Utilities`，以返回。S3URI

更改	v1	v2
Maven 依赖项	<code><dependencyManagement></code>	<code><dependencyManagement></code>

更改	v1	v2
	<pre> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> 1.12.587¹</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>s3</artifactId> </dependency> </dependencies> </pre>	<pre> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> 2.21.21²</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>s3</artifactId> </dependency> </dependencies> </pre>
软件包名称	com.amazonaws.serv ices.s3	software.amazon.aw ssdk.services.s3
类名	AmazonS3URI	S3URI

¹ [最新版本](#)。 ² [最新版本](#)。

API 变更

行为	v1	v2
解析 S3 URI。	<pre>URI uri = URI.create("https://s3.amazonaws.com"); AmazonS3Uri s3Uri = new AmazonS3URI(uri, false);</pre>	<pre>S3Client s3Client = S3Client.create(); S3Utilities s3Utilities = s3Client.utilities(); S3Uri s3Uri = s3Utilities.parseUri(uri);</pre>
从 S3 URI 中检索存储桶名称。	<pre>String bucket = s3Uri.getBucket();</pre>	<pre>Optional<String> bucket = s3Uri.bucket();</pre>
取回密钥。	<pre>String key = s3Uri.getKey();</pre>	<pre>Optional<String> key = s3Uri.key();</pre>
检索该区域。	<pre>String region = s3Uri.getRegion();</pre>	<pre>Optional<Region> region = s3Uri.region(); String region; if (s3Uri.region().isPresent()) { region = s3Uri.region().get().id(); }</pre>
检索 S3 URI 是否为路径样式。	<pre>boolean isPathStyle = s3Uri.isPathStyle();</pre>	<pre>boolean isPathStyle = s3Uri.isPathStyle();</pre>
检索版本 ID。	<pre>String versionId = s3Uri.getVersionId();</pre>	<pre>Optional<String> versionId =</pre>

行为	v1	v2
		<pre>s3Uri.firstMatchingRawQueryParameter("versionId");</pre>
检索查询参数。	不适用	<pre>Map<String, List<String>> queryParams = s3Uri.rawQueryParameters();</pre>

行为更改

网址编码

v1 提供了传入标志的选项，用于指定 URI 是否应进行网址编码。默认值为 `true`。

在 v2 中，不支持 URL 编码。如果您使用包含保留字符或不安全字符的对象密钥或查询参数，则必须对其进行 URL 编码。例如，您需要将空格替换为 " "。%20

IAM 策略生成器 API 从版本 1 到版本 2 的更改

本主题详细介绍了 IAM 策略生成器 API 从版本 1 (v1) 到版本 2 (v2) 的变化。

高级别更改

更改	v1	v2
Maven 依赖项	<pre><dependencyManagement> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> 1.12.587</version></pre>	<pre><dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> 2.21.21</version> <type>pom</ type></pre>

更改	v1	v2
	<pre> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManagem ent> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-co re</artifactId> </dependency> </dependencies> </pre>	<pre> <scope>im port</scope> </dependency> </dependencies> </dependencyManagem ent> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>iam-policy-buil der</artifactId> </dependency> </dependencies> </pre>
软件包名称	com.amazonaws.auth.policy	software.amazon.awssdk.policybuilder.iam
类名	Policy Statement <ul style="list-style-type: none"> • 声明. Effect • IdentityManagementActions • 资源 • 主体 • Condition 	IamPolicy IamStatement <ul style="list-style-type: none"> • IamEffect • IamAction • IamResource • IamPrincipal • IamCondition • IamConditionOperator • IamConditionKey

¹ [最新版本](#)。 ² [最新版本](#)。

API 变更

设置	v1	v2
实例化策略	<pre>Policy policy = new Policy();</pre>	<pre>IamPolicy.Builder policyBuilder = IamPolicy.builder(); ... IamPolicy policy = policyBuilder.buil d();</pre>
设置标识	<pre>policy.withtId(...); policy.setId(...);</pre>	<pre>policyBuilder.id(...);</pre>
设置版本	不适用-使用默认版本的 2012-10-17	<pre>policyBuilder.vers ion(...);</pre>
创建声明	<pre>Statement statement = new Statement (Effect.Allow) .withActi ons(...) .withCond itions(...) .withId(. ..) .withPrin cipals(...) .withReso urces(...);</pre>	<pre>IamStatement statement = IamStatement.build er() .effect(I amEffect.ALLOW) .actions(...) .notActio ns(...) .conditio ns(...) .sid(...) .principa ls(...) .notPrinc ipals(...) .resource s(...) .notResou rces(...)</pre>

设置	v1	v2
		<code>.build()</code>
设置陈述	<pre>policy.withStatements(statement); policy.setStatements(statement);</pre>	<pre>policyBuilder.addStatement(statement);</pre>

建立陈述的差异

操作

v1

v1 SDK 具有代表政策声明中 [Action](#) 元素的服务操作 [enum 类型](#)。以下 enum 类型是一些示例。

- [IdentityManagementActions](#)
- [DynamoDBv2Actions](#)
- [SQSActions](#)

以下示例显示了 `SendMessage` 常量 `SQSActions`。

```
Action action = SQSActions.SendMessage;
```

在 v1 中，您无法为语句指定 [NotAction](#) 元素。

v2

在 v2 中，[IamAction](#) 接口代表所有操作。要指定 [特定于服务的操作](#) 元素，请向 `create` 方法传递一个字符串，如以下代码所示。

```
IamAction action = IamAction.create("sqs:SendMessage");
```

您可以使用 v2 为 [NotAction](#) for 语句指定，如以下代码所示。

```
IamAction action = IamAction.create("sqs:SendMessage");
IamStatement.builder().addNotAction(action);
```

Conditions

v1

为了表示语句条件，v1 SDK 使用了的子类。[Condition](#)

- [ArnCondition](#)
- [BooleanCondition](#)
- [DateCondition](#)
- [IpAddressCondition](#)
- [NumericCondition](#)
- [StringCondition](#)

每个Condition子类都定义一个比较enum类型来帮助定义条件。例如，下面显示了条件的不相似[字符串比较](#)。

```
Condition condition = new StringCondition(StringComparisonType.StringNotLike, "key", "value");
```

v2

在 v2 中，您可以使用策略声明创建条件[IamConditionOperator](#)，[IamCondition](#)并提供包含所有类型的enums条件。

```
IamCondition condition = IamCondition.create(IamConditionOperator.STRING_NOT_LIKE, "key", "value");
```

资源

v1

策略声明的[Resource](#)元素由 SDK 的[Resource](#)类表示。在构造函数中将 ARN 作为字符串提供。以下子类提供了便捷的构造函数。

- [S3 BucketResource](#)
- [S3 ObjectResource](#)
- [SQS QueueResource](#)

在 v1 中，您可以[Resource](#)通过调用withIsNotType方法为指定[NotResource](#)元素，如以下语句所示。

```
Resource resource = new Resource("arn:aws:s3:::mybucket").withIsNotType(true);
```

v2

在 v2 中，您可以通过将 ARN 传递给方法来IamResource.create创建[Resource](#)元素。

```
IamResource resource = IamResource.create("arn:aws:s3:::mybucket");
```

[IamResource](#)可以将设置为[NotResource](#)元素，如以下代码段所示。

```
IamResource resource = IamResource.create("arn:aws:s3:::mybucket");  
IamStatement.builder().addNotResource(resource);
```

IamResource.ALL代表所有资源。

主体

v1

v1 SDK 提供以下[Principal](#)类来表示包含所有成员的主体类型：

- AllUsers
- AllServices
- AllWebProviders
- All

不能在语句中添加[NotPrincipal](#)元素。

v2

在 v2 中，IamPrincipal.ALL代表所有主体：

要表示其他类型的委托人中的所有成员，请在创建IamPrincipal时使用这些[IamPrincipalType](#)类。

- IamPrincipal.create(IamPrincipalType.AWS, "*")适用于所有用户。

- `IamPrincipal.create(IamPrincipalType.SERVICE, "*")` 适用于所有服务。
- `IamPrincipal.create(IamPrincipalType.FEDERATED, "*")` 适用于所有网络提供商。
- `IamPrincipal.create(IamPrincipalType.CANONICAL_USER, "*")` 适用于所有规范用户。

在创建策略声明时，您可以使用 `addNotPrincipal` 方法来表示 [NotPrincipal](#) 元素，如以下语句所示。

```
IamPrincipal principal = IamPrincipal.create(IamPrincipalType.AWS,
    "arn:aws:iam::444455556666:root");
IamStatement.builder().addNotPrincipal(principal);
```

DynamoDB APIs 映射/文档从版本 1 到版本 2 的更改

本主题详细介绍了 APIs 适用于亚马逊 DynamoDB SDK 的 Java 高级版本从版本 1.x (v1) 到 (v2) 的变化。AWS SDK for Java 2.x 我们首先介绍 object-to-table 映射 API，然后讨论 API 用于处理 JSON 样式 [文档](#) 的文档。

高级别更改

在 v1 和 v2 中，每个库中映射客户端的名称不同：

- v1-DynamoDBMapper
- v2-DynamoDB 增强版客户端

您与这两个库的交互方式大致相同：实例化映射器/客户端，然后提供一个 Java POJO 来读取这些项目并将其写入 DynamoDB 表。这两个库还为类提供了注释 POJO，以指导客户端如何处理 POJO。

迁移到 v2 时的显著区别包括：

- V2 和 v1 对低级 DynamoDB 操作使用不同的方法名称。例如：

v1	v2
load	getItem
save	putItem

v1	v2
batchLoad	batchGetItem

- V2 提供了多种定义表架构和映射POJOs到表格的方法。您可以选择使用注释或使用生成器从代码中生成的架构。V2 还提供架构的可变和不可变版本。
- 在 v2 中，您专门创建表架构作为第一步，而在 v1 中，表架构是根据需要从带注释的类中推断出来的。
- [V2 在增强版API客户端中包含 Document 客户端API，而 v1 则使用单独的客户端。API](#)
- 在 v2 中，所有版本APIs均提供同步和异步版本。

有关 v2 [增强版客户端的更多详细信息](#)，请参阅本指南中的 [DynamoDB 映射](#) 部分。

导入依赖关系

v1	v2
<pre><dependencyManagement> <dependencies> <dependency> <groupId>com.amazonaws</gro groupId> <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></pre>	<pre><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></pre>

* [最新版本](#)。

在 v1 中，单个依赖关系包括低级 DynamoDB API 和映射/文档，而在 v2 中 API，您可以使用构件依赖关系来访问映射/文档。dynamodb-enhanced API 该 dynamodb-enhanced 模块包含对低级 dynamodb 模块的传递依赖关系。

API 改变

创建客户端

应用场景	v1	v2
普通实例化	<pre>AmazonDynamoDB standardClient = AmazonDynamoDBClientBuilder.standard() .withCredentials(credentialsProvider) .withRegion(Regions.US_EAST_1) .build(); DynamoDBMapper mapper = new DynamoDBMapper(standardClient);</pre>	<pre>DynamoDbClient standardClient = DynamoDbClient.builder() .credentialsProvider(ProfileCredentialsProvider.create()) .region(Region.US_EAST_1) .build(); DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder() .dynamoDbClient(standardClient) .build();</pre>
最少的实例化	<pre>AmazonDynamoDB standardClient = AmazonDynamoDBClientBuilder.standard(); DynamoDBMapper mapper = new DynamoDBMapper(standardClient);</pre>	<pre>DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.create();</pre>
使用属性转换器 *	<pre>DynamoDBMapper mapper = new DynamoDBMapper(standardClient,</pre>	<pre>DynamoDbEnhancedClient enhancedClient</pre>

应用场景	v1	v2
	<pre>attributeTransform erInstance);</pre>	<pre>= DynamoDbEnhancedCl ient.builder() .dynamoDbClient(st andardClient) .extensions(extens ionAInstance, extensionBInstance) .build();</pre>

* v2 中的扩展与 v1 中的属性转换器大致对应。本[the section called “使用扩展”](#)节包含有关 v2 中扩展的更多信息。

建立到 DynamoDB 表/索引的映射

在 v1 中，您可以通过 Bean 注释指定 DynamoDB 表名。在 v2 中，一种工厂方法 `table()`，会生成一个代表远程 Dynam DynamoDbTable oDB 表的实例。该 `table()` 方法的第一个参数是 DynamoDB 表名。

应用场景	v1	v2
将 Java POJO 类映射到 DynamoDB 表	<pre>@DynamoDBTable(tab leName ="Customer") public class Customer { ... }</pre>	<pre>DynamoDbTable<Cust omer> customerTable = enhancedClient.tab le("Customer", TableSchema.fromBe an(Customer.class));</pre>
映射到 DynamoDB 二级索引	<ol style="list-style-type: none"> 定义一个表示索引的 POJO 类。 <ul style="list-style-type: none"> 使用 <code>@DynamoDBTable</code> 提供索引的表的名称来注释类。 可以选择使用 <code>@DynamoDBIndexHashKey</code> 和对属性进行注释。 <code>@DynamoDBIndexRangeKey</code> 	<ol style="list-style-type: none"> 用 (for a GSI) 和 <code>@DynamoDbSecondaryPartitionKey</code> <code>@DynamoDbSecondarySortKey</code> (for and or) 注释 POJO 类的 GSI 属性。 LSI 例如，

应用场景	v1	v2
	<p>2. 创建查询表达式。</p> <p>3. 使用对表示索引的POJO类的引用进行查询。例如</p> <pre>mapper.query(IdEmailIndex.class, queryExpression)</pre> <p>其中IdEmailIndex 是索引的映射类。</p> <p>《DynamoDB 开发者指南》中讨论 query v1 方法的部分显示了一个完整的示例。</p>	<pre>@DynamoDbSecondary SortKey(indexNames = "IdEmailIndex") public String getEmail() { return this.email; }</pre> <p>2. 检索对索引的引用。例如，</p> <pre>DynamoDbIndex<Customer> customerIndex = customerTable.index("IdEmailIndex");</pre> <p>3. 查询索引。</p> <p>本指南中的???部分提供了更多信息。</p>

表操作

本节介绍大多数标准用例中 v1 和 v2 之间存在差异的操作APIs。

在 v2 中，所有涉及单个表的操作都是在DynamoDbTable实例上调用的，而不是在增强型客户端上调用的。增强版客户端包含可以针对多个表的方法。

在下面名为“表操作”的表中，POJO实例被称为item或称为特定类型，例如customer1。对于 v2 示例，命名为的实例table是先前调用的结果enhancedClient.table()，它返回了对该DynamoDbTable实例的引用。

请注意，即使未显示，也可以使用流畅的使用者模式调用大多数 v2 操作。例如，

```
Customer customer = table.getItem(r # r.key(key));
    or
Customer customer = table.getItem(r # r.key(k ->
    k.partitionValue("id").sortValue("email")))
```

对于 v1 操作，该表包含一些常用的表单，而不是所有重载的表单。例如，该load()方法有以下重载：

```
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)
```

下表显示了常用的表格：

```
mapper.load(item)
mapper.load(item, config)
```

表操作

应用场景	v1	DynamoDB v2 操作	v2
向 DynamoDB 表写一个 Java POJO	<pre>mapper.save(item) mapper.save(item, config) mapper.save(item, saveExpression, config)</pre> <p>在 v1 中，DynamoDBMapperConfig.SaveBehavior 注释决定了将调用哪个低级 DynamoDB 方法。通常，UpdateItem 除非使用 SaveBehavior.CLOBBER 和，否则会被调用 SaveBehavior.PUT。自动生成的密钥是一种特殊的用例，偶尔会同时使用 PutItemUpdateItem 和。</p>	PutItemUpdateItem	<pre>table.putItem(putItemRequest) table.putItem(item) table.putItemWithResponse(item) // Returns metadata. updateItem(updateItemRequest) table.updateItem(item) table.updateItemWithResponse(item) //Returns metadata.</pre>

应用场景	v1	DynamoDB 操作	v2
从 DynamoDB 表中读取项目到 Java POJO	<pre>mapper.load(item) mapper.load(item, config)</pre>	GetItem	<pre>table.getItem(getItemRequest) table.getItem(item) table.getItem(key) table.getItemWithResponse(key) // Returns POJO with metadata.</pre>
从 DynamoDB 表中删除项目	<pre>mapper.delete(item , deleteExpression, config)</pre>	DeleteItem	<pre>table.deleteItem(deleteItemRequest) table.deleteItem(item) table.deleteItem(key)</pre>
查询 DynamoDB 表或二级索引并返回分页列表	<pre>mapper.query(Customer.class, queryExpression) mapper.query(Customer.class, queryExpression, mapperConfig)</pre>	Query	<pre>table.query(queryRequest) table.query(queryConditional)</pre> <p>使用返回的 <code>PageIterable.stream()</code> (延迟加载) 进行同步响应和 <code>PagePublisher.subscribe()</code> 异步响应</p>
查询 DynamoDB 表或二级索引并返回列表	<pre>mapper.queryPage(Customer.class, queryExpression) mapper.queryPage(Customer.class, queryExpression, mapperConfig)</pre>	Query	<pre>table.query(queryRequest) table.query(queryConditional)</pre> <p>使用返回的 <code>PageIterable.items()</code> (延迟加载) 进行同步响应和 <code>PagePublisher.items.subscribe()</code> 异步响应</p>

应用场景	v1	Dynam 操作	v2
扫描 Dynam 表或 二级 索引 并返 回分 页列 表	<pre>mapper.scan(Custom er.class, scanExpre ssion) mapper.scan(Customer .class, scanExpression, mapperConfig)</pre>	Scan	<pre>table.scan() table.scan(scanRequest)</pre> <p>使用返回的PageIterable.strea m() (延迟加载) 进行同步响应 和PagePublisher.subscribe() 异步响 应</p>
扫描 Dynam 表或 二级 索引 并返 回列 表	<pre>mapper.scanPage(Cu stomer.class, scanExpre ssion) mapper.scanPage(Cust omer.class, scanExpre ssion, mapperConfig)</pre>	Scan	<pre>table.scan() table.scan(scanRequest)</pre> <p>使用返回的PageIterable.items () (延迟加载) 进行同步响应和PagePubli sher.items.subscribe() 异步响应</p>

应用场景	v1	Dynam 操作	v2
批量读取多个表中的多个项目	<pre>mapper.batchLoad(Arrays.asList(customer1, customer2, book1)) mapper.batchLoad(itemsToGet) // itemsToGet: Map<Class<?>, List<KeyPair>></pre>	BatchItem	<pre>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.</pre>

应用场景	v1	DynamoDB 操作	v2
将多个项目批量写入多个表	<pre>mapper.batchSave(Arrays.asList(customer1, customer2, book1))</pre>	BatchWriteItem	<pre>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()))</pre>
批量删除多个表中的多个项目	<pre>mapper.batchDelete(Arrays.asList(customer1, customer2, book1))</pre>	BatchWriteItem	<pre>enhancedClient.batchWriteItem(r -> r.writeBatches(WriteBatch.builder(Record1.class) .mappedTableResource(mappedTable1) .addDeleteItem(item1key) .build(), WriteBatch.builder(Record2.class) .mappedTableResource(mappedTable2) .addDeleteItem(item2key) .build()))</pre>

应用场景	v1	DynamoDB 操作	v2
批量写入/删除多个项目	<pre>mapper.batchWrite(Arrays.asList(customer1, book1), Arrays.asList(customer2))</pre>	BatchWriteItem	<pre>enhancedClient.batchWriteItem(r -> r.writeBatches(WriteBatch.builder(Record1.class) .mappedTableResource(mappedTable1) .addPutItem(item1) .build(), WriteBatch.builder(Record2.class) .mappedTableResource(mappedTable2) .addDeleteItem(item2key) .build()))</pre>
进行事务写作	<pre>mapper.transactionWrite(transactionWriteRequest)</pre>	TransactWriteItem	<pre>enhancedClient.transactWriteItems(transactWriteItemsRequest)</pre>
进行事务性读取	<pre>mapper.transactionLoad(transactionLoadRequest)</pre>	TransactGetItem	<pre>enhancedClient.transactGetItems(transactGetItemsRequest)</pre>
获取扫描或查询中匹配项的计数	<pre>mapper.count(Customer.class, queryExpression) mapper.count(Customer.class, scanExpression)</pre>	Query Select	不支持

应用场景	v1	DynamoDB 操作	v2
在 DynamoDB 中创建与该类对应的表的 POJO	<pre>mapper.generateCreateTableRequest(Customer.class)</pre> <p>前面的语句生成低级创建表请求；用户必须在 DynamoDB createTable 客户端上调用。</p>	CreateTable	<pre>table.createTable(createTableRequest) table.createTable(r -> r.provisionedThroughput(getDefaultProvisionedThroughput()) .globalSecondaryIndices(EnhancedGlobalSecondaryIndex.builder() .indexName("gsi_1") .projection(p -> p.projectionType(ProjectionType.ALL)) .provisionedThroughput(getDefaultProvisionedThroughput()) .build()));</pre>
在 DynamoDB 中执行并行扫描	<pre>mapper.parallelScan(Customer.class, scanExpression, numTotalSegments)</pre>	Scan使用Segments数	<p>用户需要处理工作线程scan并为每个分段调用：</p> <pre>table.scan(r -> r.segment(0).totalSegments(5))</pre>

应用场景	v1	Dynam 操作	v2
将 Amazon S3 与 DynamoDB 集成以存储智能 S3 链接	<pre>mapper.createS3Link(bucket, key) mapper.getS3ClientCache()</pre>	-	不支持，因为它将 Amazon S3 和 DynamoDB 结合在一起。

地图类和属性

在 v1 和 v2 中，您都使用豆类样式注释将类映射到表。V2 还提供了[其他方法来为特定用例定义架构](#)，例如使用不可变类。

Bean 注释

下表显示了 v1 和 v2 中使用的特定用例的等效 bean 注释。Customer 类场景用于说明参数。

v2 中的注释以及类和枚举遵循驼峰大小写惯例，使用 "" 而不是 "DynamoDB"。DynamoDb

应用场景	v1	v2
将类映射到表	<pre>@DynamoDBTable (tableName = "CustomerTable")</pre>	<pre>@DynamoDbBean @dynamoDbBean(converterProviders = {...})</pre> <p>表名是在调用 DynamoDbEnhancedClient#table() 方法时定义的。</p>

应用场景	v1	v2
将类成员指定为表属性	<code>@DynamoDBAttribute (attributeName = "customerName")</code>	<code>@DynamoDbAttribute("customerName")</code>
将类成员指定为哈希键/分区键	<code>@DynamoDBHashKey</code>	<code>@DynamoDbPartitionKey</code>
将类成员指定为范围/排序键	<code>@DynamoDBHashKey</code>	<code>@DynamoDbSortKey</code>
将类成员指定为二级索引哈希/分区键	<code>@DynamoDBIndexHash Key</code>	<code>@DynamoDbSecondaryPartitionKey</code>
将类成员指定为二级索引范围/排序键	<code>@DynamoDBIndexRangeKey</code>	<code>@DynamoDbSecondarySortKey</code>
映射到表时忽略该类成员	<code>@DynamoDBIgnore</code>	<code>@DynamoDbIgnore</code>
将类成员指定为自动生成的UUID键属性	<code>@DynamoDBAutoGeneratedKey</code>	<code>@DynamoDbAutoGeneratedUuid</code> 默认情况下，不加载提供此功能的扩展；您必须将扩展程序添加到客户端生成器中。
将类成员指定为自动生成的时间戳属性	<code>@DynamoDBAutoGeneratedTimestamp</code>	<code>@DynamoDbAutoGeneratedTimestampAttribute</code> 默认情况下，不加载提供此功能的扩展；您必须将扩展程序添加到客户端生成器中。

应用场景	v1	v2
将类成员指定为自动递增的版本属性	<code>@DynamoDBVersionAttribute</code>	<code>@DynamoDbVersionAttribute</code> 提供此功能的扩展程序是自动加载的。
将类成员指定为需要自定义转换	<code>@DynamoDBTypeConverted</code>	<code>@DynamoDbConvertedBy</code>
指定要存储为其他属性类型的类成员	<code>@DynamoDBTyped(<DynamoDBAttributeType>)</code>	无等效函数
指定可以序列化为 DynamoDB 文档 JSON (样式文档) 或子文档的类	<code>@DynamoDBDocument</code>	没有直接的等效注释。使用增强版文档API。

V2 附加注释

应用场景	v1	v2
如果 Java 值为空，则指定不作为 NULL 属性存储的类成员	不适用	<code>@DynamoDbIgnoreNulls</code>
如果所有属性都为空，则将类成员指定为空对象	不适用	<code>@DynamoDbPreserveEmptyObject</code>
为班级成员指定特殊更新操作	不适用	<code>@DynamoDbUpdateBehavior</code>

应用场景	v1	v2
指定一个不可变的类	不适用	<code>@DynamoDbImmutable</code>
将类成员指定为自动递增的计数器属性	不适用	<code>@DynamoDbAtomicCounter</code> 提供此功能的扩展程序是自动加载的。

配置

在 v1 中，您通常使用实例来控制特定的行为。DynamoDBMapperConfig 您可以在创建映射器时或在发出请求时提供配置对象。在 v2 中，配置特定于操作的请求对象。

应用场景	v1	v1 中的默认值	v2
	<code>DynamoDBMapperConfig.builder()</code>		
Batch 加载重试策略	<code>.withBatchLoadRetryStrategy(batchLoadRetryStrategy)</code>	重试失败的项目	
Batch 写入重试策略	<code>.withBatchWriteRetryStrategy(batchWriteRetryStrategy)</code>	重试失败的项目	
一致性读取	<code>.withConsistentReads(CONSISTENT)</code>	EVENTUALLY	默认情况下，读取操作的一致性读取为 false。在请求对象 <code>.consistentRead(true)</code> 上使用替换。

应用场景	v1	v1 中的默认值	v2
包含编译器/解编译器的转换架构	<pre>.withConversionSchema(conversionSchema)</pre> <p>静态实现提供了与旧版本的向后兼容性。</p>	V2_COMPATIBLE	不适用。这是一项传统功能，指的是最早版本的 DynamoDB (v1) 如何存储数据类型，增强版客户端中不会保留此行为。DynamoDB v1 中的一个行为示例是将布尔值存储为数字而不是布尔值。
表名称	<pre>.withObjectContextTableNameResolver() .withTableNameOverride() .withTableNameResolver()</pre> <p>静态实现提供了与旧版本的向后兼容性</p>	使用注释或从课堂上猜测	表名是在调用 <code>DynamoDbEnhancedClient#table()</code> 方法时定义的。
分页加载策略	<pre>.withPaginationLoadingStrategy(strategy)</pre> <p>选项有：LAZY_LOADING EAGER_LOADING、或 ITERATION_ONLY</p>	LAZY_LOADING	默认为“仅迭代”。不支持其他 v1 选项。
请求收集指标	<pre>.withRequestMetricCollector(collector)</pre>	null	<code>metricPublisher()</code> 在构建标准 DynamoDB 客户端 <code>ClientOverrideConfiguration</code> 时使用。

应用场景	v1	v1 中的默认值	v2
保存行为	<pre>.withSaveBehavior(SaveBehavior.CLOBBER)</pre> <p>选项为UPDATE、CLOBBERPUT、T、或UPDATE_SKIP_NULL_ATTRIBUTES。</p>	UPDATE	<p>在 v2 中，您可以updateItem() 显式调用putItem() 或。</p> <p>CLOBBER or PUT: v 2 中的相应动作正在调用putItem()。没有特定的CLOBBER配置。</p> <p>UPDATE: 对应于 updateItem()</p> <p>UPDATE_SKIP_NULL_ATTRIBUTES : 对应于updateItem()。使用请求设置ignoreNulls 和注释 DynamoDbUpdateBehavior / 标签控制更新行为。</p> <p>APPEND_SET : 不支持</p>
类型转换器工厂	<pre>.withTypeConverterFactory(typeConverterFactory)</pre>	标准型转换器	<p>使用在 bean 上设置</p> <pre>@DynamoDbBean(converterProviders = {ConverterFactory.class, DefaultAttributeConverterProvider.class})</pre>

每个操作的配置

在 v1 中，某些操作（例如query()）可以通过提交给操作的“表达式”对象进行高度配置。例如：

```
DynamoDBQueryExpression<Customer> emailBwQueryExpr = new
DynamoDBQueryExpression<Customer>()
    .withRangeKeyCondition("Email",
        new Condition()
            .withComparisonOperator(ComparisonOperator.BEGINS_WITH)
            .withAttributeValueList(
                new AttributeValue().withS("my")));

mapper.query(Customer.class, emailBwQueryExpr);
```


在 v2 中，您不是使用配置对象，而是使用生成器在请求对象上设置参数。例如：

```
QueryEnhancedRequest emailBw = QueryEnhancedRequest.builder()
    .queryConditional(QueryConditional
        .sortBeginsWith(kb -> kb
            .sortValue("my")))
    .build();

customerTable.query(emailBw);
```

条件

在 v2 中，条件表达式和筛选表达式使用 Expression 对象来表达，该对象封装了条件以及名称和过滤器的映射。

应用场景	操作	v1	v2
预期的属性条件	保存 ()、删除 ()、查询 ()、扫描 ()	<pre>new DynamoDBS aveExpression() .withExpected(Coll ections.singletonM ap("otherAtt ribute", new ExpectedAttributeV alue(false))) .withConditionalOp erator(Conditional Operator.AND);</pre>	已弃用；ConditionExpression 改用。
条件表达式	删除 ()	<pre>deleteExpression.s etConditionExpress ion("zipcode = :zipcode") deleteExpression .setExpressionAttr ibuteValues(...)</pre>	<pre>Expression conditionExpression = Expression.builder() .expression("#key = :value OR #key1 = :value1") .putExpressionName("#key", "attribute") .putExpressionName ("#key1", "attribute3") .putExpressionValu e(":value", AttributeValues.st ringValue("wrong"))</pre>

应用场景	操作	v1	v2
			<pre> .putExpressionValue(":value1", AttributeValues.stringValue("three")) .build(); DeleteItemEnhancedRequest request = DeleteItemEnhancedRequest.builder() .conditionExpression(conditionExpression).build(); </pre>
筛选表 达式	查询 ()、扫描 ()	<pre> scanExpression .withFilterExpression("#statename = :state") .withExpressionAttributeValues(attributeValueMapBuilder.build()) .withExpressionAttributeNames(attributeNameMapBuilder.build()) </pre>	<pre> 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(); </pre>
查询条件 表达式	查询 ()	<pre> queryExpression.withKeyConditionExpression() </pre>	<pre> QueryConditional keyEqual = QueryConditional.keyEqualTo(b -> b .partitionValue("movie01")); QueryEnhancedRequest tableQuery = QueryEnhancedRequest.builder() .queryConditional(keyEqual) .build(); </pre>

类型转换

默认转换器

在 v2 中，为所有常见类型 SDK 提供了一组默认转换器。既可以在整体提供程序级别上更改类型转换器，也可以在单个属性上更改类型转换器。您可以在 [AttributeConverter](#) API 参考资料中找到可用转换器的列表。

为属性设置自定义转换器

在 v1 中，您可以使用对 getter 方法进行注释，`@DynamoDBTypeConverted` 以指定在 Java 属性类型和 DynamoDB 属性类型之间进行转换的类。例如，可以应用在 Java Currency 类型和 DynamoDB 字符串之间进行转换的，如以下代码段所示。CurrencyFormatConverter

```
@DynamoDBTypeConverted(converter = CurrencyFormatConverter.class)
public Currency getCurrency() { return currency; }
```

下面显示了上一个代码段的 v2 等效版本。

```
@DynamoDbConvertedBy(CurrencyFormatConverter.class)
public Currency getCurrency() { return currency; }
```

Note

在 v1 中，您可以将注释应用于属性本身、类型或用户定义的注释，v2 仅支持将注释应用于 getter。

添加类型转换器工厂或提供商

在 v1 中，您可以提供自己的类型转换器集，也可以通过在配置中添加类型转换器工厂来覆盖您关心的类型。类型转换器出厂扩展 `DynamoDBTypeConverterFactory`，覆盖是通过获取对默认集的引用并对其进行扩展来完成的。以下片段演示了如何执行此操作。

```
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);

```

V2 通过 `@DynamoDbBean` 注解提供了类似的功能。您可以提供单个 `AttributeConverterProvider` 或一系列订购 `AttributeConverterProvider` 的。请注意，如果您提供自己的属性转换器提供程序链，则将覆盖默认的转换器提供程序，并且必须将其包含在链中才能使用其属性转换器。

```

@dynamoDbBean(converterProviders = {
    ConverterProvider1.class,
    ConverterProvider2.class,
    DefaultAttributeConverterProvider.class})
public class Customer {
    ...
}

```

本指南中关于 [属性转换](#) 的部分包含了 v2 的完整示例。

文档历史记录API

该文档API支持在 DynamoDB 表中将JSON样式文档作为单个项目处理。v1 文档API在 v2 API 中有对应的，但是 v2 没有像 v1 API 那样为文档使用单独的客户端，而是在 DynamoDB 增强版客户端API中整合了文档功能。

在 v1 中，该 [Item](#) 类代表来自 DynamoDB 表的非结构化记录。在 v2 中，非结构化记录由该 [EnhancedDocument](#) 类的实例表示。请注意，主键是在 v2 的表架构中定义的，在 v1 的项目本身上定义的。

下表比较了 v1 和 v2 APIs 中文档之间的差异。

应用场景

v1

v2

创建文档客户端

```
AmazonDynamoDB client
= ... //Create a client.
DynamoDB documentClient
= new DynamoDB(client);
```

```
// The v2 Document API
uses the same DynamoDbE
nhancedClient
// that is used for
mapping POJOs.
DynamoDbClient
standardClient
= ... //Create a
standard client.
DynamoDbEnhancedCli
ent enhancedClient
= ... // Create an
enhanced client.
```

引用表格

```
Table documentTable
= docClient.document
Client("Person");
```

```
DynamoDbTable<Enha
ncedDocument>
documentTable =
enhancedClient.tab
le("Person",
TableSche
ma.documentSchemaB
uilder()
.addIndex
PartitionKey(Table
Metadata.primaryIn
dexName(),"id",
AttributeValueType.S)
.attribute
eConverterProvider
s(AttributeConvert
erProvider.default
Provider())
.build())
;
```

Work with semi-structured data

放置项目

```
Item item = new Item()
```

```
EnhancedDocument
personDocument =
```

应用场景

v1

```

        .withPrimaryKey("id", 50)
        .withString("firstName", "Shirley");
PutItemOutcome outcome
    = documentTable.putItem(item);

```

v2

```

EnhancedDocument.builder()
    .putNumber("id", 50)
    .putString("firstName", "Shirley")
    .build();
documentTable.putItem(personDocument);

```

获取项目

```

GetItemOutcome outcome
    = documentTable.getItemOutcome("id", 50);
Item personDocFromDb = outcome.getItem();
String firstName = personDocFromDb.getString("firstName");

```

```

EnhancedDocument personDocFromDb =
    documentTable.getItem(Key.builder()
        .partitionValue(50)
        .build());
String firstName = personDocFromDb.getString("firstName");

```

Work with JSON items

转换JSON结构以将其与文档一起使用 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();

```

放 JSON

```

documentTable.putItem(item)

```

```

documentTable.putItem(document);

```

应用场景

v1

v2

阅读 JSON

```

GetItemOutcome outcome
= //Get item.
String jsonPerson =
outcome.getItem().
toJSON();

```

```

String jsonPerson =
documentTable.getI
tem(Key.builder()
.partition
nValue(50).build())
.fromJson();

```

API文档参考和指南 APIs

	v1	v2
API参考	Javadoc	Javadoc
文档指南	Amazon DynamoDB 开发人员指南	增强文档 API (本指南)

FAQ

问：使用版本号进行乐观锁定在 v2 和 v1 中的工作方式是否相同？

答：行为类似，但是 v2 不会自动为删除操作添加条件。如果要控制删除行为，则必须手动添加条件表达式。

API从版本 1 到版本 2 的 S3 事件通知的变化

本主题详细介绍了 S3 事件通知API从版本 1.x (v1) 到版本 2.x (v2) 的更改。AWS SDK for Java

高级别更改

结构性变化

V1 对EventNotificationRecord类型及其属性使用静态内部类，而 v2 对类型使用单独的公共类。EventNotificationRecord

命名约定变更

在 v1 中，属性类名称包含后缀 Entity，而 v2 为了更简单的命名而省略了此后缀：例如，而不是 eventDataeventDataEntity。

依赖关系、包和类名的变化

在 v1 中，S3 事件通知 API 类与 S3 模块 () artifactId aws-java-sdk-s3 一起以传递方式导入。但是，在 v2 中，你需要添加对 s3-event-notifications 工件的依赖关系。

更改	v1	v2
Maven 依赖项	<pre><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> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-s3</ artifactId> </dependency> </dependencies> </dependencies></pre>	<pre><dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> 2.X.X¹</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> </dependencies></pre>

更改	v1	v2
软件包名称	<code>com.amazonaws.serv ices.s3.event</code>	<code>software.amazon.aw ssdk.eventnotifica tions.s3.model</code>

更改	v1	v2
类名	S3 EventNotification	S3 EventNotification
	S3 EventNotification .S3 EventNotificationRecord	S3 EventNotificationRecord
	S3 EventNotification。 GlacierEventDataEntity	GlacierEventData
	S3 EventNotification。 IntelligentTieringEventData Entity	IntelligentTieringEventData
	S3 EventNotification。 LifecycleEventDataEntity	LifecycleEventData
	S3 EventNotification。 ReplicationEventDataEntity	ReplicationEventData
	S3 EventNotification。 RequestParametersEntity	RequestParameters
	S3 EventNotification。 ResponseElementsEntity	ResponseElements
	S3 EventNotification。 RestoreEventDataEntity	RestoreEventData
	S3 EventNotification .S3 BucketEntity	S3Bucket
	S3 EventNotification .S3Entity	S3
	S3 EventNotification .S3 ObjectEntity	S3OBJECT
	S3 EventNotification。 TransitionEventDataEntity	TransitionEventData
		UserIdentity

更改	v1	v2
	S3 EventNotification 。 UserIdentityEntity	

¹ [最新版本](#)。

API改变

JSON转向S3EventNotification和反向

应用场景	v1	v2
S3EventNotification 从JSON字符串创建	<pre>S3EventNotification notification = S3EventNotification.parseJs on(message.body());</pre>	<pre>S3EventNotification notification = S3EventNo tification.fromJso n(message.body());</pre>
S3EventNotification 转换为JSON字符串	<pre>String json = notification.toJson();</pre>	<pre>String json = notificat ion.toJson();</pre>

的访问属性 S3EventNotification

应用场景	v1	v2
从通知中检索记录	<pre>List<S3EventNotification.S3 EventNotificationRecord> records = notifcation.getRecords();</pre>	<pre>List<S3EventNotifi cationRecord> records = notificat ion.getRecords();</pre>
从记录列表中检索记录	<pre>S3EventNotification.S3Event NotificationRecord record =</pre>	<pre>S3EventNotificatio nRecord record =</pre>

应用场景	v1	v2
	<pre>records.stream().findAny().get();</pre>	<pre>records.stream().findAny().get();</pre>
检索 Glacier 事件数据	<pre>S3EventNotification.GlacierEventDataEntity glacierEventData = record.getGlacierEventData();</pre>	<pre>GlacierEventData glacierEventData = record.getGlacierEventData();</pre>
从 Glacier 事件中检索恢复事件数据	<pre>S3EventNotification.RestoreEventDataEntity restoreEventData = glacierEventData.getRestoreEventDataEntity();</pre>	<pre>RestoreEventData restoreEventData = glacierEventData.getRestoreEventData();</pre>
检索请求参数	<pre>S3EventNotification.RequestParametersEntity requestParameters = record.getRequestParameters();</pre>	<pre>RequestParameters requestParameters = record.getRequestParameters();</pre>
检索智能分层事件数据	<pre>S3EventNotification.IntelligentTieringEventDataEntity tieringEventData = record.getIntelligentTieringEventData();</pre>	<pre>IntelligentTieringEventData intelligentTieringEventData = record.getIntelligentTieringEventData();</pre>
检索生命周期事件数据	<pre>S3EventNotification.LifecycleEventDataEntity lifecycleEventData = record.getLifecycleEventData();</pre>	<pre>LifecycleEventData lifecycleEventData = record.getLifecycleEventData();</pre>

应用场景	v1	v2
以枚举形式检索事件名称	<pre>S3EventNotification.EventNameAsString eventNameAsString = record.getEventNameAsString();</pre>	<pre>//getEventNameAsString does not exist; use 'getEventName()' String eventName = record.getEventName();</pre>
检索复制事件数据	<pre>S3EventNotification.ReplicationEventDataEntity replicationEventDataEntity = record.getReplicationEventDataEntity();</pre>	<pre>ReplicationEventData replicationEventData = record.getReplicationEventData();</pre>
检索 S3 存储桶和对象信息	<pre>S3EventNotification.S3Entity s3 = record.getS3();</pre>	<pre>S3 s3 = record.getS3();</pre>
检索用户身份信息	<pre>S3EventNotification.UserIdentityEntity userIdentityEntity = record.getUserIdentity();</pre>	<pre>UserIdentity userIdentity = record.getUserIdentity();</pre>
检索响应元素	<pre>S3EventNotification.ResponseElementsEntity responseElementsEntity = record.getResponseElements();</pre>	<pre>ResponseElements responseElements = record.getResponseElements();</pre>

S3EventNotification使用aws-lambda-java-events库进行迁移。

如果您习惯在 Lambda 函数中处理 S3 通知事件，我们建议您升级到最新的 3.x.x 版本。[aws-lambda-java-events](#)最新版本消除了 S3 事件通知API中对 AWS SDK for Java 1.x 的所有依赖关系。

并行使用适用于 Java 的 SDK 1.x 和 2.x 版

您可以在项目中同时使用 AWS SDK for Java 的两个版本。

下面显示了使用版本 1.x 中的 Amazon S3 和版本 2.16.1 中的 DynamoDB 的项目的 pom.xml 文件示例。

Example POM 示例

此示例显示了同时使用 SDK 的 1.x 和 2.x 版本的项目的 pom.xml 文件条目。

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-bom</artifactId>
      <version>1.12.1</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>bom</artifactId>
      <version>2.16.1</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>

<dependencies>
  <dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>aws-java-sdk-s3</artifactId>
  </dependency>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>dynamodb</artifactId>
  </dependency>
</dependencies>
```

适用于 AWS SDK for Java 的 OpenPGP 密钥

AWS SDK for Java 的所有公开可用的 Maven 构件均使用 OpenPGP 标准进行签名。验证构件签名所需的公有密钥可在下一部分中找到。

当前密钥

下表显示了 SDK for Java 1x 和 SDK for Java 2.x 的当前版本的 OpenPGP 密钥信息。

密钥 ID	0xAC107B386692DADD
类型	RSA
大小	4096/4096
创建时间	2016-06-30
过期时间	2024-10-08
用户 ID	AWS SDK 和工具 <aws-dr-tools@amazon.com>
密钥指纹	FEB9 209F 2F2F 3F46 6484 1E55 AC10 7B38 6692 DADD

要将以下适用于 SDK for Java 的 OpenPGP 公有密钥复制到剪贴板，请选择右上角的“复制”图标。

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
```

```
xsFNBFd1gAUBEACqbmFbxdJgz1lD7wrlskQA1LLuSAC4p8ny9u/D2zLR8Ynk3Yz
mzJuQ+Kfjne2t+xTDex6MPJlMYp0viSwsX2psgvdmeyUpW9ap0lrThNYkc+W5fRc
buFehfbi9LSATZGJi8RG0sCCr5FsYVz0gEk85M2+PeM24cXhQIOZtQUjswX/pdk/
KduGtZASqNAYLKR0mRODzUuaokLPo24pfm9bnr1RnRtw5ktPAA5bM9ZZaGKriej
kT2lPffBbjp8F5AZvmGLtNm2Cmg4FKBvI04SQjy2jjrQ3wBzi5Lc9HTxDuHK/rtV
u6PewUe2WP1nxlXenhMZU1UK4YoSB9E9StQ2VxQiySLHSdxR7Ma4WgYdVLn9b0ie
nj3QxLuQ1ZUKF79ES6JaM4t0z1gGcQeU1+UklgjFLuKwmzWRdEIFfxMyvH6qgKnd
U+DioH5mcUwhwffAAsuIJyAdMIEUYh7IfzJJXQf+fF+Xf0C16by0JFWrIGQkAzMu
CEvaCfwtHC2Lpzo33/WRFeMAuzzd0QJ4uz4xFFvaS0SZHMLHWI9YV/+Pea3X99Ms
0Nlek/LolAJh67MynHeVB0HKrq+fluorWepQivctzN6Y1N0kx5naTPGGaKWK7G2q
```

```
TbcY5SMnkIWFLFSougj0Fvmjczq8iZRwYxWA+i+LQvsR9WEXEiQffIWRoQARAQAB
zsFNBFd1gAUBEAC8zNARpWb3dPMThL2xAY+fs60vXdB1Sk0tYJpDWPfgvo0d+VQ+
hV6Xu1GAHAS6xG1WHysPT9KejIRSgLG+e9CaM5yhsxNa1WFGUM4Q9ESo3t+a75Go
7xHIxgFjC046/06Vh3g9N/PREeuG8zkZ3H2v5fmD+ejyPgk4W9sFL00zjRiZD0FK
VYR/j9uenEC/2NBcLuFy3q6cDfmCoDE0062kXMnaGz3knzEK/X1SkcjsxRDq7zaQ
lQ1Kou+3dICwy4x5SJQ8j1+eeeEvF2C2/dXmDohb57tqUwioohMUQkmCtvZgEHjy
pUwgp0MTo25gWxkvJlSJKU0b6b1786WnySIzF2gxq1kkEmB14RAssQkeXjrSmGws
MDyHNqyJeYFus18sPaSpo+V2n0z+2B070Uq+wmf1S5A5FpegH0PZzzoNZo8I6Qxa
Zje9YSZUijGmZIdEBleRVt3Svhi8MY1nasd4bW2RK1sr7plkBf8QRe6biiQRF3KD
0Sn5CbmXpAchJ1ZHRRdkXZDNQC6vCJxsy1300TrhJtAV1Yq347uyUbVi291ISVg
roUVtprismHoEk5Go0THbg9SCSt+xi/FiJQC+ubWmIGXoFKMR3UmhDnnzobKcbnbs
/Hd981FdVghYYvq//gTakJk0WxfGq030wtXRndPOA0T+qhP3TE+LtGRJ+wARAQAB
wsF1BBgBCgAPBQJXdYAFaHsMBQkHhh+AAAoJEKwQezhmktrdTyEP/0H0VHwQsaW
jMrGj000MFzXGUo8SBmYYTBs29VM8wBGDsPkYCjeZzU16i9iqDpDqxyqmTigcjH
V8CDx/6xsMBLG2yKaKZ4m3+Yn0Qf/sQkyCvqiyMF9mS7pDYWy+mPhPuw8TDIfiqg
VhzjSpIMFWPqxVjn6KKbPN/QASr3Pf0cuP6qpHG+NAM6Q5dYkCebyvwzLmg1sVni
l6iSyJd1jBj3D34XrgWS9buyxBB2CjIM76WxfNViJ9zAaPI78X9v6PpDGn0kg6oL
zrusrvBjoZknKQm0SZ+41fx6xvrTPs8uPEzevzJB1kke6kw9+KagY8mrVX1ZenRg
+sY/4vxJreYWQeq167ggx+wFjKDcfhZA7m70LH0DysrGVCLcmuinUBaN1HmLDcGY
XZ+kMCoXf0bpuCVByQmNJgEb47EIFlx/+TEeNHKM0+22xL1atFzXfkEVZck+NghL
ZyFDhS3g1bma7puU7r752uiJjA6Iv8+kHDXi+/V7GNpuiEFUYh69QQ2//CS5H51o
sC/Bkb9evSn/Lp8dMubtWAaXDGJMgw9vqZ55N02NK0fvF/IKHnGkvH28rv00PCv0
WTA/MClv28y0PrSvcmXnduLtkBEX7TISMPW+n+0Ta63/z4YFFEZ7sFLrEm3Q3vJ
MN3mE5i3cw+JGXPSu0nTtgqk/oZv//SS
=Z9u3
-----END PGP PUBLIC KEY BLOCK-----
```


文档历史记录

本主题介绍了《AWS SDK for Java 开发者指南》在其历史过程中所做的重要更改。

本指南最后一次发布于 2024 年 7 月 19 日。

更改	描述	日期
the section called “S3 事件通知”	添加一个讨论如何使用 S3 事件通知的部分 API	2024年7月19日
the section called “DynamoDB 映射/文档 APIs”	将 v1 添加到 dynamoDB 映射/文档的 v2 迁移信息 APIs	2024年7月19日
the section called “S3 事件通知”	为 S3 事件通知添加 v1 到 b2 迁移信息 API	2024年7月19日
the section called “重试”	添加重试策略主题	2024 年 6 月 18 日
如何设置 JVM TTL	使用 java 命令行系统属性删除设置networkaddress.cache.ttl 安全属性的指令。	2024 年 5 月 21 日
the section called “缩短 SDK 的启动时间 AWS Lambda”	更新HTTP客户机建议以缩短启动时间 AWS Lambda	2024 年 5 月 14 日
the section called “服务客户端指标”	重新组织指标表项目	2024 年 5 月 1 日
the section called “故障排除”	添加疑难解答主题。	2024 年 4 月 26 日
the section called “每次请求收集的指标”	添加报告的新指标SDK。	2024 年 4 月 26 日
the section called “为 DNS 名称查找设置 JVM TTL”	将推荐的DNS查找TTL时间更改为 5 秒。	2024 年 4 月 23 日
the section called “ArtifactID 映射的包名”	在 Maven artifactId 映射主题中添加软件包名称。	2024 年 4 月 17 日

更改	描述	日期
the section called “使用SDK指标”	将配置详细信息添加到指标部分。	2024 年 4 月 12 日
the section called “IAM policy 生成器 API”	添加IAM策略生成器API迁移信息。	2024 年 4 月 11 日
???	更新HTTP代理信息。	2024 年 4 月 3 日
the section called “安全地”	添加要禁用的说明IMDSv1。	2024 年 3 月 14 日
the section called “S tep-by-step 指令”	添加 step-by-step 迁移说明。	2024 年 3 月 8 日
迁移到版本 2	更新迁移主题。	2024年2月14日
the section called “配置 AWS 基于 CRT 的 HTTP 客户”	添加有关 AWS CRT基于同步的HTTP客户端的信息。	2024 年 1 月 5 日
the section called “Amazon Cognito Identity” 和 the section called “Amazon Cognito 身份提供者”	Amazon Cognito 示例移至“代码示例”部分。	2023 年 12 月 28 日
使用SDK功能	重做了SDK功能主题。	2023 年 12 月 11 日
OpenPGP 密钥	提供当前的 Op PGP en 密钥。	2023 年 12 月 6 日
the section called “序列化更改”	描述适用于 Java 的 v1 和 v2 之间的序列化差异。SDK	2023 年 12 月 5 日
the section called “S3 Transfer Manager”	添加了一个部分，详细介绍 S3 Transfer Manager 中从版本 1 到版本 2 的更改。	2023 年 11 月 13 日
the section called “注释参考”	添加了可与 DynamoDB 增强型客户端结合使用的数据类注释列表。	2023 年 10 月 30 日

更改	描述	日期
???	添加有关库和实用程序从 Java v1.x 到 v2.x SDK 的迁移状态的信息	2023 年 10 月 17 日
???	更新了 Gradle 设置主题	2023 年 10 月 17 日
the section called “忽略嵌套对象的空属性”	添加了有关 DynamoDB 增强型客户端 @DynamoDb IgnoreNulls 注释的信息。	2023 年 9 月 22 日
the section called “跨区域访问”	添加了有关对 Amazon S3 桶进行跨区域访问的信息。	2023 年 8 月 31 日
the section called “保留空对象”	添加了讨论 @DynamoDb PreserveEmptyObject 注释的部分。	2023 年 8 月 25 日
???	更新了服务客户端部分。	2023 年 8 月 15 日
the section called “客户端建议”	从 0.23 版本开始，AWS CRT 支持基于 musl 的操作系统，例如 Alpine Linux。HTTP 客户推荐现在反映了 musl 的支持。	2023 年 8 月 11 日
the section called “创建 IAM policy”	添加 IAM 策略生成器 API 部分	2023 年 7 月 31 日
the section called “开始使用”	更正了 DynamoDB 增强型客户端主题“开始使用”部分的几个代码段。	2023 年 7 月 24 日
the section called “配置 HTTP 代理”	为每个 HTTP 客户端添加 HTTP 代理支持信息和示例。	2023 年 6 月 2 日

更改	描述	日期
重新整理了目录	将 代码示例 版块和顶级条目提升与 ... 一起工作 AWS 服务 到顶级TOC条目。	2023 年 5 月 24 日
the section called “添加日志记录依赖项”	在日志部分显示 Gradle 依赖项。	2023 年 5 月 23 日
the section called “处理分页结果”	更新了分页主题。	2023 年 5 月 18 日
the section called “设置 Gradle 项目”	更新了 Gradle 项目设置。	2023 年 5 月 3 日
DynamoDB 增强版客户端 API	重写的 DynamoDB 增强版客户端主题已发布。API	2023 年 4 月 28 日
更新入门教程说明	修改了 Maven 原型以包含选项 <code>credentialsProvider</code> ；相应地修改了指令。	2023 年 4 月 11 日
the section called “客户端建议”	添加HTTP客户决策指导	2023 年 3 月 30 日
IAM最佳实践更新	更新了指南以符合IAM最佳实践。有关更多信息，请参阅 中的安全最佳实践IAM 。	2023 年 3 月 14 日
the section called “重新加载配置文件凭证”	添加了有关重新加载配置文件凭证的部分。	2023 年 2 月 9 日
the section called “配置 AWS 基于 CRT 的 HTTP 客户”	更新了有关 GA 版本的主题。	2023 年 2 月 8 日
the section called “使用 Amazon EC2 实例元数据”	添加适用于 Amazon S3 实例元数据服务的 Java SDK 客户端的指导示例。	2023 年 2 月 1 日

更改	描述	日期
the section called “使用高性能 S3 客户端”	为 AWS CRT 基于的 S3 客户端添加部分。	2022 年 12 月 19 日
the section called “传输文件和目录”	为 GA 版本更新了 Amazon S3 Transfer Manager 示例。	2022 年 12 月 19 日
the section called “最佳做法”	添加了最佳实践部分。	2022 年 11 月 18 日
the section called “从外部流程加载临时凭证”	添加了有关从外部流程加载凭证的部分。	2022 年 11 月 15 日
the section called “服务客户端指标”	更新了包含 HTTP 客户端使用要求的指标列表。	2022 年 11 月 9 日
the section called “传输文件和目录”	更正了示例代码。	2022 年 11 月 2 日
the section called “缩短 SDK 的启动时间 AWS Lambda”	更新了此部分，添加了缩短 Lambda 启动时间的选项。	2022 年 11 月 1 日
the section called “HTTP 客户”	在中添加了涵盖所有 HTTP 客户端的配置信息 SDK。	2022 年 10 月 26 日
the section called “日志记录”	更新了日志记录主题以包含所有 HTTP 客户端的线路记录详细信息。	2022 年 10 月 4 日
the section called “AWS 数据库服务”	添加了 AWS 数据库服务和 SDK 适用于 Java 2.x 的概述部分。	2022 年 9 月 13 日
EC2-经典网络即将停用	EC2-Classic 将于 2022 年 8 月 15 日退役。	2022 年 7 月 28 日
the section called “其他身份验证选项”	对单点登录身份验证所需的依赖项进行了更新。	2022 年 7 月 18 日

更改	描述	日期
the section called “传输层安全 (TLS)”	更新TLS安全信息。	2022 年 4 月 8 日
the section called “其他身份验证选项”	添加了有关设置和使用凭证的更多信息。	2021 年 2 月 22 日
the section called “设置 GraalVM 原生映像项目”	添加了设置 GraalVM 原生映像项目的新主题。	2021 年 2 月 18 日
the section called “轮询资源状态”	发布了 Waiter；为新功能添加了主题。	2020 年 9 月 30 日
the section called “使用SDK指标”	发布了指标；为新功能添加了主题。	2020 年 8 月 17 日
the section called “Amazon SNS”	添加了的示例主题 Amazon SNS。	2020 年 5 月 30 日
the section called “缩短 SDK 的启动时间 AWS Lambda”	添加了 AWS Lambda 函数性能主题。	2020 年 5 月 29 日
the section called “为 DNS 名称查找设置 JVM TTL”	添加了JVMTTLDNS缓存主题。	2020 年 4 月 27 日
the section called “设置 Apache Maven 项目”, the section called “设置 Gradle 项目”	新增 Maven 和 Gradle 设置主题。	2020 年 4 月 21 日
the section called “传输层安全 (TLS)”	在安全部分中添加了 TLS 1.2。	2020 年 3 月 19 日
the section called “订阅 Amazon Kinesis Data Streams”	添加了 Kinesis 直播示例。	2018 年 8 月 2 日

更改	描述	日期
the section called “处理分页结果”	添加了自动分页主题。	2018 年 4 月 5 日
???	添加了 IAM Amazon EC2、CloudWatch 和的示例主题 DynamoDB。	2017 年 12 月 29 日
the section called “Amazon S3”	添加了 getObject 的示例。 Amazon S3	2017 年 7 月 8 日
the section called “使用异步编程”	添加了异步主题。	2017 年 8 月 4 日
AWS SDK for Java 2.x 的 GA 版本	AWS SDK for Java 版本 2 (v2) 已发布。	2017 年 6 月 28 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。