版本 2.x 开发人员指南

AWS SDK for Java 2.x



Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS SDK for Java 2.x: 版本 2.x 开发人员指南

Copyright © 2026 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

什么是 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 3
设置 4
设置概述 4
安装 Java 开发先决条件 5
设置 Apache Maven 项目 5
设置 Gradle 项目 11
设置 GraalVM 原生映像项目 17
使用进行身份验证 AWS 19
身份验证的设置 19
其他身份验证选项 21
创建简单的应用程序 21
步骤 1：为本教程进行设置 21
步骤 2：创建项目 21
步骤 3：编写代码 26
步骤 4：构建并运行应用程序 30
后续步骤 31
配置服务客户端 32
通过外部方式配置客户端 33
用于客户端配置的配置提供程序链 33
创建使用外部设置配置的服务客户端 34
适用于 Java 的 SDK 2.x 环境变量和 JVM 系统属性 34
代码中的客户端配置 35
代码中的基本配置 35
代码中的高级配置 36
代码中不可用的配置选项 37
单例服务客户端 38
单例服务客户端的好处 39
创建和使用单例服务客户端 39

重要注意事项 39
AWS 区域 40
显式配置 AWS 区域 40
根据环境确定区域 41
查看服务可用性 42
选择特定端点 43
凭证提供程序 43
交互式开发工作 44
默认凭证提供程序链 46
凭证缓存 50
指定特定的凭证提供程序 53
使用共享配置文件 55
使用外部进程 57
在代码中提供凭证 62
在亚马逊上阅读 IAM 角色证书 EC2 65
重试 67
重试策略 67
指定策略 71
自定义策略 72
从 RetryPolicy 迁移到 RetryStrategy 74
超时 74
服务客户端超时 74
HTTP 客户端超时 76
超时交互和层次结构 78
使用智能配置默认值 79
摘要 79
可观测性 80
指标 80
监控 108
日志记录 108
了解如何查看、监控和管理 SageMaker 端点。 118
端点配置选项 118
代码内端点配置 119
请求级别端点配置 120
外部端点配置 120
配置优先级 122

特定于服务的端点配置 122
最佳实践 123
配置 HTTP 客户端 124
可用客户端 124
客户端建议 125
智能默认值 128
配置基于 Apache 的 HTTP 客户端 130
配置基于 URLConnection 的 HTTP 客户端 135
配置基于 Netty 的 HTTP 客户端 141
配置基于 AWS CRT 的 HTTP 客户端 147
配置 HTTP 代理 160
配置基于 Apache 5.x 的 HTTP 客户端 165
拦截器 167
生命周期 167
注册拦截器 168
示例 168
最佳实践 176
上下文对象 176
使用 SDK 177
发出 AWS 服务请求 180
使用服务客户端发出请求 180
提出请求 181
处理回复 183
异步编程 184
使用异步客户端 API 184
使用异步方法处理流 187
配置高级异步选项 191
最佳实践 192
防止请求挂起 192
重复使用客户端 193
发布资源 193
防止出现连接问题 193
HTTP 优化 193
SSL 优化 194
监控性能 194
处理错误 194

为什么使用未选中的异常？ 194
AwsServiceException（和子类） 195
SdkClientException 196
异常和重试行为 196
分页 196
同步分页 196
异步分页 199
Waiter 204
先决条件 204
使用 Waiter 205
配置 Waiter 206
代码示例 206
故障排除 207
连接重置 207
连接超时 207
读取超时 208
连接池超时 208
类路径错误 211
签名错误 212
连接池关闭 214
凭证提供程序链错误 215
为 AWS Lambda 缩短 SDK 启动时间 218
使用基于 AWS CRT 的 HTTP 客户端 218
移除未使用的 HTTP 客户端依赖项 218
配置服务客户端以进行快捷查找 219
在 Lambda 函数处理程序之外初始化 SDK 客户端 220
尽量减少依赖关系注入 221
使用针对 AWS Lambda 的 Maven 原型 221
Lambda SnapStart 221
影响启动时间的 2.x 版更改 221
其他资源 222
实现ContentStreamProvider 222
使用 mark() 和 reset() 222
如果 mark() 和 reset() 不可用，则使用缓冲 223
创建新流 223
为 DNS 名称查找设置 JVM TTL 223

如何设置 JVM TTL 224
使用 HTTP/2 224
调用 AWS 服务 226
CloudWatch 226
从中获取指标 CloudWatch 227
将自定义指标数据发布到 CloudWatch 228
处理 CloudWatch 警报 230
使用 Amazon CloudWatch Events 234
AWS 数据库服务 238
Amazon DynamoDB 238
Amazon RDS 239
Amazon Redshift 239
Amazon Aurora Serverless v1 240
Amazon DocumentDB 240
DynamoDB 240
选择您的 DynamoDB 客户端 240
配置 DynamoDB 客户端 241
本主题涵盖的内容 242
处理 DynamoDB 中的表 242
处理中的项目 DynamoDB 251
将对象映射到 DynamoDB 项目 259
Amazon EC2 386
管理 Amazon EC2 实例 386
使用 AWS 区域 区和可用区 393
在 Amazon EC2 中使用安全组 400
使用 Amazon EC2 实例元数据 405
IAM 410
管理 IAM 访问密钥 411
管理 IAM 用户 416
创建 IAM policy 421
使用 IAM 策略 429
使用 IAM 服务器证书 435
Kinesis 439
订阅 Amazon Kinesis Data Streams 440
Lambda 450
调用 Lambda 函数 450

列出 Lambda 函数 452
删除 Lambda 函数 452
Amazon S3 453
SDK 中的 S3 客户端 453
将流上传到 S3 456
预签名 URL 462
跨区域访问 471
使用校验和实现数据完整性保护 472
使用高性能 S3 客户端 479
配置并行传输支持 483
传输文件和目录 485
S3 事件通知 494
Amazon SNS 504
创建主题 504
列出 Amazon SNS 主题 505
为终端节点订阅主题 506
向主题发布消息 507
为终端节点取消订阅主题 508
删除主题 509
Amazon SQS 510
使用自动请求批处理 510
队列操作 515
消息操作 518
Amazon Transcribe 521
设置麦克风 522
创建发布者 522
创建客户端和启动流 525
更多信息 521
代码示例 527
ACM 530
操作 530
API Gateway 548
操作 530
场景 552
AWS 社区捐款 553
Application Auto Scaling 554

操作 530
应用程序恢复控制器 562
操作 530
Aurora 565
开始使用 565
基本功能 566
操作 530
场景 552
Auto Scaling 600
开始使用 565
基本功能 566
操作 530
场景 552
AWS Batch 664
开始使用 565
基本功能 566
操作 530
Amazon Bedrock 713
操作 530
Amazon Bedrock 运行时系统 718
场景 552
Amazon Nova 736
Amazon Nova Canvas 756
Amazon Titan 图像生成器 759
Amazon Titan 文本嵌入 762
Anthropic Claude 765
Cohere Command 784
Meta Llama 794
Mistral AI 805
Stable Diffusion 816
CloudFront 819
操作 530
场景 552
CloudWatch 848
开始使用 565
基本功能 566

操作 530
场景 552
CloudWatch 活动 923
操作 530
CloudWatch 日志 929
操作 530
场景 552
Amazon Cognito Identity 948
操作 530
Amazon Cognito 身份提供者 955
开始使用 565
操作 530
场景 552
Amazon Comprehend 982
操作 530
场景 552
AWS Control Tower 994
开始使用 565
基本功能 566
操作 530
Firehose 1055
操作 530
场景 552
Amazon DocumentDB 1065
无服务器示例 1065
DynamoDB 1066
开始使用 565
基本功能 566
操作 530
场景 552
无服务器示例 1065
AWS 社区捐款 553
亚马逊 EC2 1326
开始使用 565
基本功能 566
操作 530

场景 552
Amazon ECR 1423
开始使用 565
基本功能 566
操作 530
Amazon ECS 1463
操作 530
ELB – 版本 2 1477
开始使用 565
操作 530
场景 552
MediaStore 1521
操作 530
AWS Entity Resolution 数据匹配服务 1536
开始使用 565
基本功能 566
操作 530
OpenSearch 服务 1583
开始使用 565
基本功能 566
操作 530
EventBridge 1613
开始使用 565
基本功能 566
操作 530
场景 552
EventBridge 调度器 1648
开始使用 565
操作 530
场景 552
预测 1672
操作 530
Amazon Glacier 1685
操作 530
AWS Glue 1701
开始使用 565

基本功能 566
操作 530
HealthImaging 1733
操作 530
场景 552
IAM 1763
开始使用 565
基本功能 566
操作 530
场景 552
Amazon Inspector 1847
开始使用 565
基本功能 566
操作 530
AWS IoT 1895
开始使用 565
基本功能 566
操作 530
AWS IoT data 1933
操作 530
AWS IoT FleetWise 1937
开始使用 565
基本功能 566
操作 530
AWS IoT SiteWise 1990
开始使用 565
基本功能 566
操作 530
Amazon Keyspaces 2028
开始使用 565
基本功能 566
操作 530
Kinesis 2054
操作 530
无服务器示例 1065
AWS KMS 2067

开始使用 565
基本功能 566
操作 530
Lambda 2122
开始使用 565
基本功能 566
操作 530
场景 552
无服务器示例 1065
AWS 社区捐款 553
Amazon Lex 2159
场景 552
Amazon Location 2159
开始使用 565
基本功能 566
操作 530
Location Service Places 2206
操作 530
AWS Marketplace 目录 API 2211
AMI 产品 2212
渠道合作伙伴优惠 2237
容器产品 2253
实体 2259
优惠 2264
产品 2323
转售授权 2328
SaaS 产品 2369
公用事业机构 2395
AWS Marketplace 协议 API 2399
协议 2399
MediaConvert 2452
操作 530
Migration Hub 2473
操作 530
Amazon MSK 2486
无服务器示例 1065

Neptune 2487
开始使用 565
基本功能 566
操作 530
场景 552
合作伙伴中心 2529
操作 530
场景 552
Amazon Personalize 2564
操作 530
Amazon Personalize Events 2594
操作 530
Amazon Personalize Runtime 2597
操作 530
Amazon Pinpoint 2602
操作 530
Amazon Pinpoint SMS 和 Voice API 2646
操作 530
Amazon Polly 2649
操作 530
场景 552
Amazon RDS 2656
开始使用 565
基本功能 566
操作 530
场景 552
无服务器示例 1065
Amazon RDS 数据服务 2698
场景 552
Amazon Redshift 2699
开始使用 565
基本功能 566
操作 530
场景 552
Amazon Rekognition 2738
操作 530

场景 552
Route 53 域注册 2809
开始使用 565
基本功能 566
操作 530
Amazon S3 2832
开始使用 565
基本功能 566
操作 530
场景 552
无服务器示例 1065
Amazon S3 控件 3017
开始使用 565
基本功能 566
操作 530
S3 目录存储桶 3061
开始使用 565
基本功能 566
操作 530
场景 552
SageMaker AI 3142
开始使用 565
操作 530
场景 552
Secrets Manager 3171
操作 530
Amazon SES 3173
操作 530
场景 552
Amazon SES API v2 3189
操作 530
场景 552
Amazon SNS 3209
开始使用 565
操作 530
场景 552

无服务器示例 1065
Amazon SQS 3277
开始使用 565
操作 530
场景 552
无服务器示例 1065
Step Functions 3375
开始使用 565
基本功能 566
操作 530
场景 552
AWS STS 3399
操作 530
支持 3402
开始使用 565
基本功能 566
操作 530
Systems Manager 3425
开始使用 565
基本功能 566
操作 530
Amazon Textract 3468
操作 530
场景 552
Amazon Transcribe 3480
操作 530
场景 552
Amazon Transcribe Streaming 3491
操作 530
场景 552
Amazon Translate 3510
场景 552
迁移到版本 2 3512
版本 2 中有哪些新功能 3512
如何迁移 3513
迁移工具 3513

分步说明 3523
1.x 和 2.x 之间的差异 3540
软件包名称更改 3540
将版本 2.x 添加到您的项目 3541
不可变 POJO 3542
Setter 和 getter 方法 3542
模型类名 3543
库和实用工具 3543
客户端更改 3545
凭证提供程序更改 3590
区域更改 3598
操作、请求和响应更改 3600
Exception 更改 3609
特定于服务的更改 3610
使用 Amazon S3 3616
配置文件更改 3662
外部配置 3663
Waiter 3666
EC2 元数据实用程序 3670
CloudFront 预签名 3678
IAM policy 生成器 API 3681
使用 DynamoDB 3687
SQS 自动请求批处理 3720
并行使用适用于 Java 的 SDK 1.x 和 2.x 版 3729
查找使用 1.x 客户端的应用程序 3730
使用 CloudTrail Lake 查找具有 1.x 客户端的应用程序 3730
安全性 3732
数据保护 3732
传输层安全性协议（TLS） 3733
检查 TLS 版本 3733
强制执行 TLS 版本 3734
迁移到 TLS 1.2 3734
身份和访问管理 3734
受众 3735
使用身份进行身份验证 3735
使用策略管理访问 3736

如何 AWS 服务 使用 IAM 3738
对 AWS 身份和访问进行故障排除 3738
合规性验证 3740
恢复能力 3740
基础设施安全性 3740
OpenPGP 密钥 3742
当前密钥 3742
以前的密钥 3747
文档历史记录 3754
..... mmmdcclxiv

什么是 AWS SDK for Java 2.x

为 AWS SDK for Java 提供了 Java API AWS 服务。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。

AWS SDK for Java 2.x 这是对 1.x 版本代码库的重大改写。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验 SDK，请按照入门教程进行操作。

要设置开发环境，请参阅the section called “设置”。

如果您当前使用的是 1.x 版 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 和其他提出请求的信息 AWS 服务，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，请 Amazon Web Services 提供AWS Amplify框架。

SDK 主要版本的维护和支持

有关 SDK 主要版本及其底层依赖项的维护和支持的信息，请参阅AWS SDKs 和工具参考指南中的以下主题：

- AWS SDKs 和工具维护政策
- AWS SDKs 和工具版本 Support Matrix

其他资源

除本指南外，以下是 AWS SDK for Java 开发人员宝贵的在线资源：

- AWS SDK for Java 2.x API 参考
- Java 开发人员博客
- 中的 Java 开发主题 AWS re:Post
- SDK 源已开启 GitHub
- 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：

- 在上@@ 提交 SDK 问题 GitHub
- 在 AWS SDK for Java 2.x g itter 频道上加入关于 SDK 的非正式聊天

AWS SDK for Java 2.x 入门

本主题中的各个部分将引导您完成开始构建连接到 AWS 服务 的 Java 应用程序的基本步骤。这些部分介绍了使用 Java 和构建工具（如 Maven 或 Gradle）设置开发环境、配置向 AWS 进行安全身份验证，以及通过一个动手实践的教程创建您的第一个可运行的应用程序。这个适合初学者的主题是您使用 Java 进行 AWS 开发的切入点，为您后续探索更高级的功能打下所需的基础。

目录

- 设置AWS SDK for Java 2.x
  - 设置概述
  - 安装 Java 和构建工具以与AWS SDK for Java 2.x 配合使用
  - 设置使用AWS SDK for Java 2.x 的 Apache Maven 项目
    - 先决条件
    - 创建 Maven 项目
    - 为 Maven 配置 Java 编译器
    - 将开发工具包声明为依赖项
    - 设置开发工具包模块的依赖项
      - 将整个开发工具包构建到您的项目中
    - 重新构建项目。
  - 设置使用AWS SDK for Java 2.x 的 Gradle 项目
  - 设置使用AWS SDK for Java 2.x 的 GraalVM 原生映像项目
    - 先决条件
    - 使用原型创建项目
    - 生成原生映像
- AWS 使用进行身份验证 AWS SDK for Java 2.x
  - 身份验证的设置
    - 1. 使用短期证书进行本地开发
    - 2. 设置用于 SDK 的单点登录访问
    - 3. 使用登录 AWS CLI
  - 其他身份验证选项
- 使用AWS SDK for Java 2.x 创建简单的应用程序

- 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
  - 成功
  - 清理
- 后续步骤

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

要使用AWS SDK for Java 成功开发访问 AWS 服务的应用程序，需要满足以下条件：

- Java SDK 必须有权访问凭证才能代表您对请求进行身份验证。
- 为 SDK 配置的 IAM 角色的权限必须提供您的应用程序需访问的 AWS 服务的访问权限。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：
    - `config` 文件包含 IAM Identity Center 单点登录设置，以便 SDK 可以获取 AWS 凭证。
    - `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）一个 IDE（集成开发环境），例如 IntelliJ IDEA、Eclipse 或 NetBeans。

    如果您使用 IntelliJ IDEA，也可以添加 AWS Toolkit for IntelliJ IDEA 以将 AWS 服务 直接集成到 IDE，从而帮助您简化开发。

- 准备好运行应用程序时，具有活动的 AWS 访问门户会话。您可以使用 AWS Command Line Interface 来启动登录流程，从而登录 IAM Identity Center 的 AWS 访问门户。

> **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。

安装 Java 和构建工具以与AWS SDK for Java 2.x 配合使用

要使用适用于 Java 的 SDK 2.x，您需要满足以下 Java 开发环境要求：

- Java 8 或更高版本。AWS SDK for Java 可以与 Oracle Java SE 开发套件以及 Amazon Corretto、Red Hat OpenJDK 和 Adoptium 等开放 Java 开发套件（OpenJDK）的发行版配合使用。
- 与 Maven 兼容的构建工具，例如 Apache Maven、Apache Ant with Ivy、Gradle 或 IntelliJ。
  - 有关如何安装和使用 Maven 的信息，请访问 https://maven.apache.org/。
  - 有关如何安装和使用 Apache Ivy 的信息，请访问 https://ant.apache.org/ivy/。
  - 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/。

设置使用AWS SDK for Java 2.x 的 Apache Maven 项目

您可以使用 Apache Maven 设置和构建 AWS SDK for Java 2.x 项目或构建 SDK 本身。

先决条件

要将适用于 Java 的 SDK 2.x 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。适用于 Java 的 SDK 2.x 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，`pom.xml`）中打开 `myapp` 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 .jar 目录（例如，target）中创建单个 myapp/target 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。

设置使用AWS SDK for Java 2.x 的 Gradle 项目

您可以使用 Gradle 来设置和构建AWS SDK for Java 2.x 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:33.3.0-jre")
}

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

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

```
}

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

4. 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:2.27.21"))
       // With the bom declared, you specify individual SDK dependencies without a
    version.
    ...
   }
   ...
   ```

   > **Note**
   >
   > 在此示例构建文件中，将 2.27.21 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:2.27.21"))
       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:ssooidc")
    implementation(platform("org.apache.logging.log4j:log4j-bom:2.20.0"))
    implementation("org.apache.logging.log4j:log4j-slf4j2-impl")
    implementation("org.apache.logging.log4j:log4j-1.2-api")
    testImplementation(platform("org.junit:junit-bom:5.10.0"))
    testImplementation("org.junit.jupiter:junit-jupiter")
}

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

```
}

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

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

Groovy DSL (build.gradle)

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

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

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

dependencies {
    implementation platform('software.amazon.awssdk:bom:2.27.21')
    implementation 'software.amazon.awssdk:s3'
    implementation 'software.amazon.awssdk:sso'
    implementation 'software.amazon.awssdk:ssooidc'
    implementation platform('org.apache.logging.log4j:log4j-bom:2.20.0')
    implementation 'org.apache.logging.log4j:log4j-slf4j2-impl'
    implementation 'org.apache.logging.log4j:log4j-1.2-api'
```

```
    testImplementation platform('org.junit:junit-bom:5.10.0')
    testImplementation 'org.junit.jupiter:junit-jupiter'
}

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

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

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

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。

设置使用AWS SDK for Java 2.x 的 GraalVM 原生映像项目

在 2.16.1 及更高版本中，AWS SDK for Java 2.x 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=2.27.21\
    -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 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。

AWS 使用进行身份验证 AWS SDK for Java 2.x

当你使用 AWS 服务 AWS SDK for Java 2.x 进行开发 AWS 时，你必须确定如何进行身份验证。SDK 完全在幕后管理凭证发现、签名创建和凭证刷新，让您可以专注于应用程序逻辑。

身份验证的设置

《和工具参考指南》中的身份验证 AWS SDKs 和访问主题描述了不同的身份验证方法。

对于本地开发，推荐的方法是使用您的控制台登录凭证登录 AWS CLI，从而使用短期证书。如果您使用根、IAM 用户或与 IAM 的联合身份验证进行 AWS 账户访问，则建议您这样做。按照 使用 AWS SDK for Java 2.x访问交互式开发工作的凭证 中的说明进行操作。

按照 AWS SDKs 和《工具参考指南》中的说明进行操作后，您的系统应设置为允许 SDK 对请求进行签名：

1. 使用短期证书进行本地开发

对于本地开发，推荐的方法是使用您的控制台登录凭证登录 AWS CLI，从而使用短期证书。如果您使用根、IAM 用户或与 IAM 的联合身份验证进行 AWS 账户访问，则建议您这样做。

有关说明，请参阅中的使用凭证提供程序 AWS SDK for Java 2.x。

2. 设置用于 SDK 的单点登录访问

在完成编程访问部分的步骤 2 以使 SDK 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。`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使用的 AWS 服务 。

  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。`sso-session` 节包含启动 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 和工具参考指南中的 SSO 令牌提供者配置。

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

3. 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`。

要测试您是否已有活动会话，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。

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 SDKs 和工具参考指南中的配置章节。

使用AWS SDK for Java 2.x 创建简单的应用程序

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：

- Amazon S3 访问权限
- 一个 Java 开发环境，配置为使用 AWS IAM Identity Center 单点登录访问 AWS 服务

请按照 ??? 中的说明为本教程进行设置。在为 Java SDK 将开发环境配置为单点登录访问，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=2.27.21
```

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>2.27.21</aws.java.sdk.version> <-------- SDK version
picked up from archetype version.
        <slf4j.version>1.7.28</slf4j.version>
        <junit5.version>5.8.1</junit5.version>
    </properties>

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

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

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

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

```
            <artifactId>ssooidc</artifactId> <-------- Required for identity center
authentication.
        </dependency>

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

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

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

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

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

```
    </dependencies>

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

</project>
```

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.waiter().waitUntilBucketExists(HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build());
            System.out.println(bucketName + " is ready.");
            System.out.printf("%n");
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void cleanUp(S3Client s3Client, String bucketName, String keyName) {
        System.out.println("Cleaning up...");
        try {
            System.out.println("Deleting object: " + keyName);
            DeleteObjectRequest deleteObjectRequest =
  DeleteObjectRequest.builder().bucket(bucketName).key(keyName).build();
            s3Client.deleteObject(deleteObjectRequest);
```

```
            System.out.println(keyName + " has been deleted.");
            System.out.println("Deleting bucket: " + bucketName);
            DeleteBucketRequest deleteBucketRequest =
 DeleteBucketRequest.builder().bucket(bucketName).build();
            s3Client.deleteBucket(deleteBucketRequest);
            System.out.println(bucketName + " has been deleted.");
            System.out.printf("%n");
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Cleanup complete");
        System.out.printf("%n");
    }
}
```

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：

- 使用 Amazon S3
- 使用其他 Amazon Web Services（例如DynamoDB、Amazon EC2 和各种数据库服务）
- 使用 SDK
- AWS SDK for Java 的安全性

在 AWS SDK for Java 2.x中配置服务客户端

为了以编程方式进行访问 AWS 服务，适用于 Java 的 SDK 2.x 使用一个客户端对象作为每个对象。AWS 服务例如，如果您的应用程序需要访问亚马逊 EC2，则可以创建一个亚马逊 EC2客户端对象（Ec2Client 类的实例）来与该服务交互。然后，您可以使用服务客户端向该 AWS 服务发出请求。对于大多数应用程序，您可以使用服务客户端的单例实例。

配置 SDK 行为的方法有很多，但归根结底，一切都与服务客户端的行为有关。在您的代码创建使用配置的服务客户端之前，任何配置都不会生效。

您提供的配置示例包括：

- 当您调用服务 AWS 时，您的代码如何进行身份验证
- AWS 区域 你想让服务客户端使用的
- 服务调用的重试和超时设置
- HTTP 代理配置

请参阅AWS SDKs 和工具参考指南，了解设置、功能和其他常见的基本概念。 AWS SDKs

主题

- 通过外部方式为AWS SDK for Java 2.x 配置服务客户端
- 在代码中为AWS SDK for Java 2.x 配置服务客户端
- 将单例服务客户端实例与AWS SDK for Java 2.x 一起使用
- 为AWS SDK for Java 2.x 设置 AWS 区域
- 在中使用凭证提供程序 AWS SDK for Java 2.x
- 在AWS SDK for Java 2.x 中配置重试行为
- 在AWS SDK for Java 2.x 中配置超时
- 在AWS SDK for Java 2.x 中配置可观测性功能
- 在AWS SDK for Java 2.x 中配置客户端端点
- 在中配置 HTTP 客户端 AWS SDK for Java 2.x
- 在AWS SDK for Java 2.x 中使用执行拦截器

通过外部方式为AWS SDK for Java 2.x 配置服务客户端

许多配置设置可以通过代码以外的方式来处理。当您通过外部方式处理配置时，配置可以在同一个 Java 进程中应用于所有应用程序。大多数配置设置可以设置为环境变量、JVM 系统属性或在单独的共享 AWS `config` 文件中设置。共享 `config` 文件可以维护多组独立的设置（称为配置文件），以便为不同的环境或测试提供不同的配置。

大多数环境变量和共享 `config` 文件设置都已标准化，可在 AWS SDK 和工具之间共享，从而支持在不同编程语言和应用程序之间保持一致的功能。在大多数情况下，适用于 Java 的 SDK 可以使用的 JVM 系统属性与环境变量一一对应。

请参阅《AWS SDK 和工具参考指南》，了解如何通过这些方法配置应用程序，以及每个跨 SDK 设置的详细信息。要查看 SDK 可以从环境变量、JVM 系统属性或配置文件中解析的所有设置，请参阅《AWS SDK 和工具参考指南》中的设置参考。

用于客户端配置的配置提供程序链

SDK 会检查几个位置（或来源）以查找配置值。

1. 在代码中或服务客户端本身设置的任何显式设置均优先于其他任何设置。
2. JVM 系统属性
   - 有关设置 JVM 系统属性的详细信息，请参阅《AWS SDK 和工具参考指南中的如何设置 JVM 系统属性。
3. 环境变量
   - 有关设置环境变量的详细信息，请参阅《AWS SDK 和工具参考指南》中的环境变量。
   - 请注意，您可以在不同作用域层级为 shell 配置环境变量：系统级、用户级，以及特定终端会话级。
4. 共享 `config` 文件和 `credentials` 文件
   - 有关设置这些文件的详细信息，请参阅《AWS SDK 和工具参考指南》中的共享 `config` 和 `credentials` 文件。
5. 最后使用 SDK 源代码本身提供的任何默认值。
   - 某些属性（例如“Region”）没有默认值。您必须在代码、环境设置或共享 `config` 文件中明确指定这些属性。如果 SDK 无法解析所需的配置，API 请求在运行时会失败。

除了这个通用配置链之外，适用于 Java 的 SDK 2.x 还使用专门的提供程序链，包括凭证提供程序链和 AWS 区域提供程序链。这些专门的链会添加其他提供程序，这些提供程序会考虑 SDK 的运行环境。例如，在容器或 EC2 实例中。

创建使用外部设置配置的服务客户端

您需要在应用程序中创建一个服务客户端才能与 AWS 服务通信。服务客户端是您与 AWS 服务建立连接的关键组件，它会处理所有复杂的通信细节，让您无需为此操心。它们会自动处理安全性、错误处理和重试等重要任务，让您可以专注于构建应用程序，而不是处理复杂的技术问题。

使用 `create()` 方法

如果您需要的所有配置设置都来自外部来源，则可以使用简单方法创建服务客户端：

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

前面的代码段创建了一个 `S3Client` 实例。在创建过程中，SDK 会按照配置提供程序链依次查找所需的设置。SDK 找到某个设置值后，便会立即采用，即使链中的后续位置还存在该配置，也会忽略后续配置。

例如，假设用户通过设置系统属性 `-Daws.region=us-west-2` 来设置 AWS 区域的 JVM 设置。如果还设置了 `AWS_REGION` 环境变量，则其值将被忽略。

创建过程中还将使用默认区域提供程序链和默认凭证提供程序链。在链的某个地方，SDK 必须解析要使用的 AWS 区域，并找到使其能够为签名请求检索凭证的设置。如果 SDK 无法找到这些值，则客户端创建失败。

尽管您可以使用这个空的生成器模式来创建客户端，但当您想在代码中添加配置时，通常会使用这种模式。

适用于 Java 的 SDK 2.x 环境变量和 JVM 系统属性

除了大多数 AWS SDK 支持的跨 SDK 设置外，适用于 Java 的 SDK 2.x 还提供以下设置。

> **Note**
>
> 这些环境变量和 JVM 系统属性主要用于高级使用案例、测试或特定部署场景。在大多数应用程序代码中，最好使用 SDK 的客户端生成器提供的编程配置选项，以便获得更好的类型安全性和 IDE 支持。

容器凭证提供程序环境变量

除了参考指南中记录的标准容器凭证环境变量外，SDK 还支持：

AWS_CONTAINER_SERVICE_ENDPOINT – 使用容器凭证提供程序时，此环境变量指定容器元数据服务的端点。

Java 系统属性：aws.containerServiceEndpoint

默认值：http://169.254.170.2

HTTP 客户端实现环境变量

SYNC_HTTP_SERVICE_IMPL – 明确标识 SDK 将使用的默认同步 HTTP 实现。当类路径上有多个实现，这种做法非常有用；因为实现发现需要对类路径进行扫描，这也是一种性能优化手段。

Java 系统属性：software.amazon.awssdk.http.service.impl

ASYNC_HTTP_SERVICE_IMPL – 明确标识 SDK 将使用的默认异步 HTTP 实现。当类路径上有多个实现，这种做法非常有用；因为实现发现需要对类路径进行扫描，这也是一种性能优化手段。

Java 系统属性：software.amazon.awssdk.http.async.service.impl

在代码中为AWS SDK for Java 2.x 配置服务客户端

除了（或作为替代方案）通过外部方式配置服务客户端之外，您还可以在代码中以编程方式对其进行配置。

通过在代码中配置服务客户端，您可以精细地控制许多可用的选项。可以通过外部方式设置的大多数配置也可以在代码中进行设置。

代码中的基本配置

例如，以下代码段在代码中为 Amazon S3 服务客户端将 AWS 区域设置为 EU_SOUTH_2：

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

前面的代码段显示了静态工厂方法 builder()。builder() 方法返回一个允许您自定义服务客户端的 builder 对象。fluent setter 方法会返回 builder 对象（在本例中为 S3ClientBuilder 实

例），这样就可以将多个方法调用链式地写在一起，使代码更简洁、更易读。在配置了所需属性后，可以调用 build() 方法创建客户端。

代码中的高级配置

下面的代码段显示了额外配置选项：

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

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

在前面的代码段中，您可以看到配置服务客户端的几个入口点：

- 提供所有服务客户端通用的配置选项的 **ClientOverrideConfiguration.Builder 对象**。这些设置是与任何 HTTP 实现无关的 AWS 特定行为。
- 通过单独的 HTTP 客户端生成器实现来配置 HTTP 客户端。ApacheHttpClient.Builder 就是一个这样的例子。服务客户端提供 httpClientBuilder() 方法，将配置好的 HTTP 客户端关联到服务客户端。
- 客户端生成器本身的方法，例如 region() 和 credentialsProvider()

使用配置块来实现相同的配置

不是使用先创建单独的对象，然后再将对象传递给服务客户端这样的方法，AWS SDK for Java 2.x 提供的方法可接受 lambda 表达式来内联地构建这些对象。生成器上的配置方法名称相同，但具有不同的签名。例如：

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

前面所示的 S3 客户端配置，如果使用这种方法，可以用一个代码块来完成：

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

代码中不可用的配置选项

由于以下设置会影响 SDK 中的基本初始化过程，因此只能通过外部方式而不能在代码中设置以下配置设置：

文件位置设置

这些设置控制共享配置和凭证文件的位置，在 SDK 加载这些设置后无法以编程方式对其进行覆盖：

- AWS_CONFIG_FILE（环境变量）/aws.configFile（JVM 系统属性）
- AWS_SHARED_CREDENTIALS_FILE（环境变量）/aws.sharedCredentialsFile（JVM 系统属性）

因为这些设置决定了 SDK 在哪里查找配置，所以必须在 SDK 加载配置文件之前进行设置。SDK 初始化完成后，更改这些值将不再生效。

实例元数据服务禁用

- AWS_EC2_METADATA_DISABLED（环境变量）/aws.disableEc2Metadata（JVM 系统属性）

此设置用于控制 SDK 是否会尝试使用 EC2 实例元数据服务。SDK 初始化完成后，就无法以编程方式更改此设置。

配置文件选择

- AWS_PROFILE（环境变量）/aws.profile（JVM 系统属性）

此设置告诉 SDK 从共享配置和凭证文件中加载哪个配置文件。加载后，更改此值将不再生效。

容器凭证路径

- AWS_CONTAINER_CREDENTIALS_RELATIVE_URI
- AWS_CONTAINER_CREDENTIALS_FULL_URI
- AWS_CONTAINER_AUTHORIZATION_TOKEN
- AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE

使用这些环境变量告诉 SDK 如何从容器服务获取凭证。在服务客户端初始化期间建立凭证提供程序链后，无法再更改这些设置。

默认 HTTP 实现选择

- SYNC_HTTP_SERVICE_IMPL（环境变量）/software.amazon.awssdk.http.service.impl（JVM 系统属性）
- ASYNC_HTTP_SERVICE_IMPL（环境变量）/software.amazon.awssdk.http.async.service.impl（JVM 系统属性）

这些全局设置决定了 SDK 在所有服务客户端中默认使用哪个 HTTP 客户端实现，除非在代码中为单个服务客户端覆盖了该设置。您必须在 SDK 初始化其 HTTP 客户端之前进行设置，之后不能再更改。

将单例服务客户端实例与AWS SDK for Java 2.x 一起使用

AWS SDK for Java 2.x 中的服务客户端是线程安全的。您可以为每个服务客户端创建一个实例，并在整个应用程序中重复使用该实例。这种方法可以提高性能并更有效地管理资源。

单例服务客户端的好处

连接池

服务客户端维护内部 HTTP 连接池。创建和销毁这些池的成本很高。当您重复使用客户端时，可以在不同请求之间高效共享这些池。

降低了初始化开销

创建客户端涉及加载配置、建立凭证和初始化内部组件。使用单例实例可消除这种开销。

更高效地利用资源

使用单例客户端可防止因为创建许多客户端实例而耗尽资源。

创建和使用单例服务客户端

以下示例说明了如何创建和使用单例服务客户端：

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

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

不要为每个操作创建新客户端：

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

重要注意事项

- 服务客户端是线程安全的。您可以安全地在多个线程中共享客户端。

- 仅在应用程序关闭或不再需要客户端时才关闭客户端。在应用程序级别使用 `client.close()` 或 try-with-resources。
- 如果您需要不同的配置（例如区域或凭证），请为每个配置创建单独的单例实例。

如果您使用像 Spring 这样的依赖项注入框架，请将服务客户端配置为单例 bean。这样可以确保适当的生命周期管理。

为AWS SDK for Java 2.x 设置 AWS 区域

SDK 客户端会连接到您在创建客户端时指定的特定 AWS 区域中的 AWS 服务。借助此配置，您的应用程序可以与该地理区域中的 AWS 资源进行交互。当您在未明确设置区域的情况下创建服务客户端时，SDK 将使用外部配置中的默认区域。

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**
>
> 使用生成器所构建的客户端不可改变，而且 AWS 区域不能更改。如果您需要为同一项服务使用多个 AWS 区域，请创建多个客户端，即每个区域一个客户端。

让 SDK 根据环境自动确定默认 AWS 区域

当您的代码在 Amazon EC2 或 AWS Lambda 上运行时，您可能需要将客户端配置为与所运行代码使用同一个 AWS 区域。由此可以将代码从其运行的环境中脱离，更轻松地将应用程序部署到多个 AWS 区域，以减少延迟并保证冗余。

要使用默认的 AWS 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();
```

您也可以用其他方式配置客户端，但不能设置区域。SDK 使用默认区域提供程序链来获取 AWS 区域：

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

如果您没有使用 `region` 方法显式设置 AWS 区域，SDK 将参考默认区域提供程序链来确定要使用的区域。

了解默认 AWS 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` 方法明确设置的所有区域优先于任何其他区域。
2. SDK 会查找 JVM 系统属性 `aws.region`，如果找到则使用其值。
3. 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**
   >
   > 该环境变量通过 Lambda 容器设置。

4. SDK 会检查 AWS 共享配置和凭证文件中的活动配置文件。如果 `region` 属性存在，则 SDK 会使用该属性。

   除非被 `AWS_PROFILE` 环境变量或 `aws.profile` JVM 系统属性覆盖，否则 `default` 配置文件是活动配置文件。如果 SDK 在两个文件中找到同一个配置文件（包括 `default` 配置文件）的 `region` 属性，则 SDK 将使用共享凭证文件中的值。

5. SDK 将尝试使用 Amazon EC2 实例元数据服务（IMDS），为当前运行的 Amazon EC2 实例确定区域。
   - 为了提高安全性，您应该禁止 SDK 尝试使用 IMDS 的版本 1。您可以使用 the section called “安全地” 部分中描述的相同设置来禁用版本 1。
6. 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 应用程序的一个常用方法是，使用共享配置文件在本地开发时设置区域，而在 AWS 基础设施上运行应用程序时依赖默认区域提供程序链来确定区域。这可以明显简化客户端创建，并保证应用程序的便携性。

检查某项服务在某个区域内是否可用

要确认特定的 AWS 服务在某个区域内是否可用，请对服务客户端使用静态 `serviceMetadata` 方法：

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

前面的代码段输出一长串带有 DynamoDB 服务的 AWS 区域代码：

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

您可以使用代码来查找您希望服务客户端使用的区域的区域类枚举。

例如，如果您想在代码为 `ap-northeast-2` 的区域内使用 DynamoDB，请至少使用以下配置来创建 DynamoDB 客户端：

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

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 客户端配置为使用欧洲地区（爱尔兰）区域并指定特定端点，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端点的最新列表，请参阅区域和端点。

在中使用凭证提供程序 AWS SDK for Java 2.x

证书提供商在中的角色 AWS SDK for Java 2.x 是为 SDK 的 AWS 服务 客户端获取和提供凭证。SDK 使用其获取的凭证对每个请求进行加密签名，从而向服务进行身份验证。凭证通常包含访问密钥（一个访问密钥 ID 和一个秘密访问密钥）。

当您使用临时证书（在设置 SSO 令牌提供商配置或将运行时配置为担任 IAM (AWS Identity and Access Management) 角色时使用）时，会话令牌会添加到访问密钥中，从而提供对资源的限时访问权限。 AWS

本主题讨论了使 SDK 能够访问凭证的几种方式。

主题

- 使用 AWS SDK for Java 2.x访问交互式开发工作的凭证
- AWS SDK for Java 2.x中的默认凭证提供程序链
- 证书缓存在 AWS SDK for Java 2.x
- 在中指定特定的凭证提供商 AWS SDK for Java 2.x
- 在中使用 AWS 共享配置文件 AWS SDK for Java 2.x
- 使用 AWS SDK for Java 2.x从外部进程加载凭证
- 使用 AWS SDK for Java 2.x在代码中提供凭证
- EC2 使用适用于 Java 的 SDK 2.x 在亚马逊上读取 IAM 角色证书

使用 AWS SDK for Java 2.x访问交互式开发工作的凭证

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。

您可以通过几种方法来使用临时凭证。您使用的方法以及您向 SDK 提供的配置都取决于您的使用案例。

使用 Java SDK 进行交互式开发工作时，我们建议您使用 AWS 控制台登录凭据。

使用控制台登录凭证

您可以使用现有的 AWS 管理控制台登录凭据以编程方式访问 AWS 服务。在基于浏览器的身份验证流程之后， AWS 生成可在本地开发工具（如 AWS CLI 和 SDK for Java 2.x）上使用的临时证书。

通过此流程，您可以使用在初始账户设置期间创建的根证书、IAM 用户或身份提供商提供的联合身份进行身份验证，然后 AWS CLI 自动为您管理临时证书。这种方法无需在本地存储长期凭证，从而增强了安全性。

运行 `aws login` 命令时，您可以从活动控制台会话中进行选择，也可以通过基于浏览器的身份验证流程登录，这将自动生成临时凭证。适用于 Java 的 SDK 2.x 将在最长 12 小时内自动刷新这些凭证。

> **Important**
>
> 除了您在共享配置文件中设置的适用于所有项目的配置外，每个 Java 项目都需要在 Maven `pom.xml` 文件中具有以下依赖关系：
>
> ```
> <dependency>
>     <groupId>software.amazon.awssdk</groupId>
>     <artifactId>signin</artifactId>
> </dependency>
> ```
>
> 该`signin`依赖项提供了允许适用于 Java 2.x 的 SDK 访问和使用控制台登录凭据的代码。

有关先决条件、登录和注销的更多信息，请参阅 AWS SDK 和工具参考指南中的使用控制台凭据登录进行 AWS 本地开发。

Single-sign-on 方法

使用 Java SDK 进行交互式开发工作时，也可以使用单点登录方法。此方法需要以下设置：

- 通过 IAM Identity Center 设置
- 在 AWS 共享配置文件中配置配置文件
- 使用 AWS CLI 并运行命令登录并创建活动会话

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 角色并检索用于签署 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 AWS SDKs 和工具参考指南的了解 IAM Identity Center 身份验证部分。

> **Important**
>
> 除了在共享 `config` 文件中设置的适用于所有项目的配置外，每个 Java 项目还需要在 Maven `pom.xml` 文件中具有以下依赖项：
>
> ```
> <dependency>
>     <groupId>software.amazon.awssdk</groupId>
>     <artifactId>sso</artifactId>
> </dependency>
> <dependency>
>     <groupId>software.amazon.awssdk</groupId>
>     <artifactId>ssooidc</artifactId>
> </dependency>
> ```
>
> `sso` 和 `ssooidc` 依赖项提供的代码使适用于 Java 的 SDK 2.x 能够访问临时凭证。

从 AWS 访问门户检索临时证书

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 2，为可授予访问权限以满足开发需求的 IAM 角色名称选择 `Access keys`。此角色的名称通常类似于PowerUserAccess或开发人员。
   b. 在模态对话框中，选择您的操作系统，然后从将配置文件添加到您的 AWS 凭证文件复制内容。
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
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
aws_session_token=IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZ2luX2IQoJb3JpZVERYLONG
```

7. 保存 `credentials` 文件。

当适用于 Java 的 SDK 创建服务客户端时，它将访问这些临时凭证并将它们用于每个请求。在步骤 5a 中选择的 IAM 角色的设置决定了临时凭证的有效时间。最长持续时间为 12 小时。

在临时凭证过期后，重复步骤 4 到 7。

AWS SDK for Java 2.x中的默认凭证提供程序链

中的默认凭证提供程序链 AWS SDK for Java 2.x 会自动在预定义的位置序列中搜索 AWS 凭据，从而允许应用程序在 AWS 服务 不明确指定凭据来源的情况下进行身份验证。

默认凭证提供程序链由该DefaultCredentialsProvider类实现。它会按顺序将凭证获取任务委托给其他凭证提供程序实现，这些实现会依次检查不同位置的配置。出现第一个可以找到所有必要配置元素的凭证提供程序后，链就会终止。

要使用默认凭证提供程序链来提供临时凭证，请创建服务客户端生成器，但不要指定凭证提供程序。以下代码段创建一个 DynamoDbClient，使用默认凭证提供程序链来查找和检索配置设置。

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

凭证设置检索顺序

适用于 Java 的 SDK 2.x 的默认凭证提供程序链使用预定义的顺序在您的环境中搜索配置。

1. Java 系统属性
   - 软件开发工具包使用SystemPropertyCredentialsProvider类从aws.accessKeyIdaws.secretAccessKey、和 aws.sessionToken Java 系统属性加载临时证书。

     > **Note**
     >
     > 有关如何设置 Java 系统属性的信息，请参阅官方 Java Tutorials 网站中的 System Properties 教程。

2. 环境变量
   - SDK 使用EnvironmentVariableCredentialsProvider类从AWS_ACCESS_KEY_IDAWS_SECRET_ACCESS_KEY、和AWS_SESSION_TOKEN环境变量加载临时证书。
3. Web 身份令牌和 IAM 角色 ARN
   - SDK 使用该WebIdentityTokenFileCredentialsProvider类通过使用 Web 身份令牌扮演角色来加载凭证。
   - 凭证提供程序会查找以下环境变量或 JVM 系统属性：
     - AWS_WEB_IDENTITY_TOKEN_FILE or aws.webIdentityTokenFile
     - AWS_ROLE_ARN 或 aws.roleArn
     - AWS_ROLE_SESSION_NAME 或 aws.roleSessionName（可选）

- SDK 获取值后，它会调用 AWS Security Token Service (STS) 并使用返回的临时证书对请求进行签名。
- 诸如 Amazon Elastic Kubernetes Service (EKS) 之类的运行环境会自动 AWS SDKs向其提供网络身份令牌，从而使应用程序能够获得临时证书。 AWS

4. 共享 `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 SDKs 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 AWS 服务临时证书。

  > **Note**
  >
  > `credentials`和`config`文件由各种 AWS SDKs 和工具共享。有关更多信息，请参阅。_ aws/credentials and .aws/config AWS SDKs 和《工具参考指南》中的文件。

- 由于共享 `credentials` 和 `config` 文件中的配置文件包含许多不同的设置集，因此 `ProfileCredentialsProvider` 会委托给一系列其他提供程序，以便在 `[default]` 配置文件下查找设置：
  - Web 身份令牌凭证（类 `WebIdentityTokenCredentialsProvider`）：当配置文件包含 `role_arn` 和 `web_identity_token_file` 时。
  - SSO 凭证（类 `SsoCredentialsProvider`）：当配置文件包含与 SSO 相关的属性（例如 `sso_role_name`、`sso_account_id`）时。
  - 带有来源配置文件且基于角色的凭证（类 `StsAssumeRoleCredentialsProvider`）：当配置文件包含 `role_arn` 和 `source_profile` 时。
  - 带有凭证来源且基于角色的凭证（类 `StsAssumeRoleWithSourceCredentialsProvider`）：当配置文件包含 `role_arn` 和 `credential_source` 时。
    - 当 `credential_source = Environment` 时：使用 `SystemPropertyCredentialsProvider` 和 `EnvironmentVariableCredentialsProvider` 组成的链
    - 当 `credential_source = Ec2InstanceMetadata` 时：使用 `InstanceProfileCredentialsProvider`
    - 当 `credential_source = EcsContainer` 时：使用 `ContainerCredentialsProvider`

- 控制台登录凭证（类别LoginCredentialsProvider）：当配置文件包含时 login_session
- 进程凭证（类 ProcessCredentialsProvider）：当配置文件包含 credential_process 时。
- 会话凭证（类 StaticSessionCredentialsProvider）：当配置文件包含 aws_access_key_id、aws_secret_access_key 和 aws_session_token 时。
- 基本凭证（类 StaticCredentialsProvider）：当配置文件包含 aws_access_key_id 和 aws_secret_access_key 时。

5. Amazon ECS 容器凭证
   - SDK 使用ContainerCredentialsProvider类通过以下环境变量加载临时证书：
     a. AWS_CONTAINER_CREDENTIALS_RELATIVE_URI 或 AWS_CONTAINER_CREDENTIALS_FULL_URI
     b. AWS_CONTAINER_AUTHORIZATION_TOKEN_FILE 或 AWS_CONTAINER_AUTHORIZATION_TOKEN

   ECS 容器代理会自动设置指向 ECS 凭证端点的 AWS_CONTAINER_CREDENTIALS_RELATIVE_URI 环境变量。在不使用标准 ECS 凭证端点的特定场景中通常会设置其他环境变量。
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。
7. 如果 SDK 无法通过上面列出的所有步骤找到必要的配置设置，则会抛出异常，输出类似于以下内容：

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

在代码中使用 **DefaultCredentialsProvider**

您可以在代码中显式使用默认凭证提供程序链。因为 SDK 默认使用 DefaultCredentialsProvider，这在功能上等同于您根本不指定凭证提供程序。但显式使用默认凭证提供程序链可以使代码更具可读性，并且能够自我说明。这清楚地表明了您打算使用默认凭证链。

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

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

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

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

在构建默认凭证提供程序时，您可以提供更多配置：

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

这种方法为您提供了更多控制，同时仍然保留默认凭证链的便利性。

证书缓存在 AWS SDK for Java 2.x

AWS SDK for Java 2.x 实现凭据缓存以提高性能并减少对凭证源的调用。本节说明凭证缓存的工作原理以及如何为应用程序配置凭证缓存。

了解凭证提供程序缓存

适用于 Java 的 SDK 2.x 中的凭证提供程序使用不同的缓存策略：

- 内部凭证缓存：许多提供程序会缓存其检索到的凭证。
- 自动刷新：具有缓存凭证的提供程序会实施刷新机制。

具有内部凭证缓存的提供程序

即使创建了新实例，以下凭证提供程序也会在内部缓存凭证：

- 实例配置文件凭证提供者：缓存来自 Amazon EC2 元数据服务的证书。
- 容器凭证提供程序：缓存来自容器元数据端点的凭证。
- 基于 STS 的提供商：缓存来自 AWS Security Token Service (STS) 的临时证书。
- Web 身份令牌提供程序：缓存从 Web 身份令牌获取的凭证。
- 进程凭证提供程序：缓存来自外部进程的凭证。

没有内部缓存的提供程序

以下提供程序不实施内部缓存：

- 环境变量凭证提供程序
- 系统属性凭证提供程序
- 静态凭证提供程序

配置凭证缓存

在构建凭证提供程序时，您可以自定义缓存行为：

过时时间

控制凭证何时被视为过时且需要刷新：

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

预取时间

确定在凭证过期之前何时开始刷新：

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

异步更新

启用后台凭证刷新：

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

会话持续时间

对于基于 STS 的提供程序，控制临时凭证的有效时长：

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

缓存凭证配置示例

举例来说，为凭证提供程序实现配置缓存时，您可能希望让 SDK 使用后台线程在凭证过期之前预取（提前检索）凭证。这样，您就可以避免因检索新凭证而导致调用阻塞。

以下示例创建了一个 StsAssumeRoleCredentialsProvider，通过在生成器上将 asyncCredentialUpdateEnabled 属性设置为 `true`，使用后台线程来预取凭证：

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

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

首次在上`s3Client`调用操作时，会向 AWS Security Token Service (STS) 发送一个`AssumeRoleRequest`。STS 返回有效期为 15 分钟（900 秒）的临时凭证。`s3Client` 实例使用缓存的凭证，直到在 15 分钟到期之前需要刷新凭证为止。默认情况下，SDK 会在当前会话到期前 5

分钟到 1 分钟之间尝试为新会话检索新凭证。预取窗口可使用 prefetchTime 和 staleTime 属性进行配置。

您可以用类似的方式配置以下基于会话的凭证提供程序：

- StsWebIdentityTokenFileCredentialsProvider
- StsGetSessionTokenCredentialsProvider
- StsGetFederationTokenCredentialsProvider
- StsAssumeRoleWithWebIdentityCredentialsProvider
- StsAssumeRoleWithSamlCredentialsProvider
- StsAssumeRoleCredentialsProvider
- DefaultCredentialsProvider（当委托给使用会话的凭证提供程序时）
- ProcessCredentialsProvider
- WebIdentityTokenFileCredentialsProvider
- ContainerCredentialsProvider
- InstanceProfileCredentialsProvider

了解凭据缓存有助于您在使用时优化应用程序的性能和可靠性。 AWS 服务

在中指定特定的凭证提供商 AWS SDK for Java 2.x

虽然默认凭证提供程序链在许多情况下都很方便，但明确指定凭证提供程序可以让您更好地控制身份验证行为、性能和安全性。

您可能想要指定凭证提供程序的原因可能包括：

- 默认提供程序链按顺序检查多个来源，这样会增加延迟：

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

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

- 您需要使用非标准位置来访问凭证配置：

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

- 为不同服务客户端使用不同的凭证。例如，如果您的应用程序需要访问多个 AWS 账户或对不同的服务使用不同的权限：

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

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

- 控制凭证刷新行为：

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

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

```
    .build();
```

在中使用 AWS 共享配置文件 AWS SDK for Java 2.x

使用共享 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、InstanceProfileCredent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.
*/
```

使用 AWS SDK for Java 2.x从外部进程加载凭证

> **Warning**
>
> 以下内容描述从外部进程获取临时凭证的方法。这可能很危险，因此请谨慎行事。我们建议：只要有可能，就尽量使用其他凭证提供程序。如果使用此选项，我们建议您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中写入任何秘密信息StdErr。 SDKs 并且 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《 AWS SDKs 和工具参考指南》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();
```

使用 IAM Roles Anywhere 进行身份验证

IAM Roles An AWS 服务 ywher e 允许您获取在外部运行的工作负载的临时 AWS 证书 AWS。借助该服务，您可以从本地或其他云环境安全访问 AWS 资源。

在使用 IAM Roles Anywhere 对请求进行身份验证之前，您首先需要收集所需的信息并下载凭证助手工具。按照《IAM Roles Anywhere 用户指南》中的入门说明进行操作，您可以创建必要的构件。

适用于 Java 的 SDK 没有专门的凭证提供程序来从 IAM Roles Anywhere 检索临时凭证，但您可以使用凭证助手工具以及其中一个选项从外部进程检索凭证。

在配置文件中使用 **credential_process** 设置

共享 AWS 配置文件中的以下片段显示了使用该`credential_process`设置的名为`roles_anywhere`的配置文件：

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

组装完所有构件后，您需要用您的值替换以红色显示的文本。设置中的第一个元素 `aws_signing_helper` 是凭证助手工具的可执行文件，`credential-process` 是命令。

当您将服务客户端配置为使用 `roles_anywhere` 配置文件时（如以下代码所示），SDK 会缓存临时凭证并在临时凭证到期之前更新凭证：

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

配置 **ProcessCredentialsProvider**

如下所示，您可以使用仅代码方法，使用 `ProcessCredentialsProvider`，而不是使用配置文件设置：

```
ProcessCredentialsProvider processCredentialsProvider =
 ProcessCredentialsProvider.builder()
    .command("""
            ./aws_signing_helper credential-process \
            --certificate /path/to/certificate \
            --private-key /path/to/private-key \
            --trust-anchor-arn arn:aws:rolesanywhere:region:account:trust-anchor/TA_ID
 \
            --profile-arn arn:aws:rolesanywhere:region:account:profile/PROFILE_ID \
            --role-arn arn:aws:iam::account:role/role-name-with-path
        """).build();

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

组装完所有构件后，用您的值替换以红色显示的文本。

使用 AWS SDK for Java 2.x在代码中提供凭证

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

中定义的以下权限集 AWS IAM Identity Center 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。

JSON

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

代入的角色

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存储桶。

JSON

```
{
    "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。

JSON

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

EC2 使用适用于 Java 的 SDK 2.x 在亚马逊上读取 IAM 角色证书

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 A mazon EC2 实例上运行的应用程序授予权限。

本主题提供有关如何将 Java 应用程序设置为在 EC2实例上运行以及 AWS SDK for Java 2.x 如何允许获取 IAM 角色证书的信息。

从环境中获取 IAM 角色凭证

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会在执行环境中搜索配置元素，SDK 可以用其换取临时凭证。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，软件开发工具包使用`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 角色凭证

作为最终使用 `InstanceProfileCredentialsProvider` on 的默认凭证提供程序链的替代方案 EC2，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 角色凭证

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访问已配置的 IAM 角色等信息。`InstanceProfileCredentialsProvider` EC2 默认情况下，实例运行两个版本的 IMDS：

- 实例元数据服务版本 1 (IMDSv1)-一种 request/response 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。更安全的方法 IMDSv1。

默认情况下，Java SDK 首先 IMDSv2 尝试获取 IAM 角色，但如果失败，则会尝试 IMDSv1。但是，由于 IMDSv1 不太安全，因此 AWS 建议 IMDSv2 仅使用并禁用 SDK 进行尝试 IMDSv1。

要使用更安全的方法，请提供以下设置之一，禁用 SDK 的使用 IMDSv1 ，其值为`true`。

- 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

IMDSv1 如果其中一个设置设置为true，则在初始 IMDSv2 调用失败时，SDK 不会通过使用来加载 IMDS 角色凭证。

在AWS SDK for Java 2.x 中配置重试行为

对 AWS 服务的调用偶尔会由于意想不到的原因而失败。某些错误 [例如节流（超出允许的速率限制）或临时性错误] 是暂时的，如果重试调用，可能会成功。AWS SDK for Java 2.x 具有内置机制，可检测此类错误并自动重试所有客户端默认启用的调用。

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 SDK 中用于实现重试的一种机制。每个 SDK 客户端在构建时都会生成一个重试策略，该策略在客户端构建完成后就无法再修改。

重试策略负责完成以下任务。

- 将异常归类为可重试或不可重试。
- 计算下一次重试之前建议等待的延迟时间。
- 维护一个令牌存储桶，该存储桶提供了一种机制，可以在很大一部分请求失败且重试也失败时停止重试。

> **Note**
>
> 在 SDK 版本 2.26.0 发布重试 strategy（策略）之前，SDK 中的重试机制由重试 policy（策略）提供。重试 policy（策略） API 由 software.amazon.awssdk.core.retry 软件包中的核心 RetryPolicy 类组成，而 software.amazon.awssdk.retries 软件包包含重试 strategy（策略） API 元素。
> 重试策略（retry strategy）API 的引入，是 AWS 为统一各 SDK 核心组件的接口和行为而作出的全局努力的一部分。

适用于 Java 的 SDK 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、服务端节流、并发或乐观锁失败以及暂时性服务错误。

标准重试策略

对于正常使用案例，标准重试策略是推荐的 RetryStrategy 实现。与 AdaptiveRetryStrategy 不同，标准策略通常适用于所有重试使用案例。

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。您可以使用 StandardRetryStrategy.Builder#retryOnException 进行调整。
- 重试 2 次，总共 3 次尝试。您可以使用 StandardRetryStrategy.Builder#maxAttempts(int) 进行调整。
- 对于非节流异常，使用 BackoffStrategy#exponentialDelay 退避策略，基本延迟为 100 毫秒，最大延迟为 20 秒。您可以使用 StandardRetryStrategy.Builder#backoffStrategy 进行调整。
- 对于节流异常，使用 BackoffStrategy#exponentialDelay 退避策略，基本延迟为 1 秒，最大延迟为 20 秒。您可以使用 StandardRetryStrategy.Builder#throttlingBackoffStrategy 进行调整。
- 在下游服务频繁失败的情况下，执行断路（禁用重试）。始终执行第一次尝试，仅禁用重试。使用 StandardRetryStrategy.Builder#circuitBreakerEnabled 进行调整。

传统重试策略

传统重试策略是 RetryStrategy，适用于普通使用案例，但它已被弃用，取而代之的是 StandardRetryStrategy。当您未指定另一个策略时，这是客户端使用的默认重试策略。

该策略的特点是对节流和非节流异常采取不同的处理方式，对于节流异常，退避的基本延迟（500 毫秒）大于非节流异常的基本延迟（100 毫秒），并且节流异常不会影响令牌存储桶状态。

在 AWS 内部大规模使用该策略的经验表明，该策略并不比标准重试策略表现得更好。此外，该策略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匮乏。

默认情况下，传统重试策略执行以下操作。

- 根据构建时配置的条件进行重试。您可以使用 LegacyRetryStrategy.Builder#retryOnException 进行调整。

- 重试 3 次，总共 4 次尝试。您可以使用 `LegacyRetryStrategy.Builder#maxAttempts(int)` 进行调整。
- 对于非节流异常，使用 `BackoffStrategy#exponentialDelay` 退避策略，基本延迟为 100 毫秒，最大延迟为 20 秒。您可以使用 `LegacyRetryStrategy.Builder#backoffStrategy`. 进行调整。
- 对于节流异常，使用 `BackoffStrategy#exponentialDelay` 退避策略，基本延迟为 500 毫秒，最大延迟为 20 秒。您可以使用 `LegacyRetryStrategy.Builder#throttlingBackoffStrategy` 进行调整。
- 在下游服务频繁失败的情况下，执行断路（禁用重试）。断路永远不会阻止成功的第一次尝试。您可以通过 `LegacyRetryStrategy.Builder#circuitBreakerEnabled` 调整此行为。
- 断路器的状态不受节流异常的影响。

自适应重试策略

自适应重试策略是 `RetryStrategy`，适用于资源限制程度高的使用案例。

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，用于衡量节流请求相较于非节流请求的比率。该策略使用此衡量标准来减慢请求速度，目的是尽量保持在安全的带宽范围内，理想情况下可完全避免触发节流错误。

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。您可以使用 `AdaptiveRetryStrategy.Builder#retryOnException` 进行调整。
- 重试 2 次，总共 3 次尝试。您可以使用 `AdaptiveRetryStrategy.Builder#maxAttempts(int)` 进行调整。
- 使用动态的退避延迟，该延迟根据下游资源当前的负载动态调整。
- 在下游服务出现大量失败时，执行断路（禁用重试）。断路可能会阻止在中断情况下进行第二次尝试，从而保护下游服务。

> **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处理多个资源，则当客户端访问所有其他资源时，与一个资源相关的节流或中断会导致延迟增加和出现故障。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。
在所有客户端都对资源使用自适应重试策略的情况下，我们建议您也使用此策略。

Important

Java SDK 2.26.0 发布的重试策略包括新的 `RetryMode.ADAPTIVE_V2` 枚举值。ADAPTIVE_V2 模式修复了一个错误：当之前已检测到节流错误时，未能使第一次尝试延迟。
在 2.26.0 版本中，用户可以通过环境变量、系统属性或配置文件设置将模式设置为 `adaptive`，从而自动获得 ADAPTIVE_V2 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 `RetryMode.ADAPTIVE` 在代码中设置模式来获取之前的行为。

摘要：重试策略默认值的比较

下表显示每种重试策略的属性的默认值。

策略	最大尝试次数	非节流错误的基本延迟	节流错误的基本延迟	令牌存储桶大小	每个非节流重试的令牌成本	每个节流重试的令牌成本
Standard	3	100 毫秒	1000 毫秒	500	5	5
传统	4	100 毫秒	500 毫秒	500	5	0
自适应	3	100 毫秒	100 毫秒	500	5	5

Note

对于所有重试策略，DynamoDB 客户端使用默认的最大重试次数为 8，该值高于上表中为其他 AWS 服务客户端显示的值。

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` 来代替 `RetryMode.STANDARD`。

作为配置文件设置

包括 `retry_mode` 作为共享 AWS 配置文件中的配置文件设置。指定 `standard`、`legacy` 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

您可以使用系统属性 `aws.retryMode` 为所有服务客户端配置重试策略，除非在代码中覆盖此设置。指定 `standard`、`legacy` 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

您也可以使用值为 standard、legacy 或 adaptive 的 AWS_RETRY_MODE 环境变量。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 会话的重试模式设置为 standard。

```
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已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，您可以生成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多 3 次尝试替换为 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**
>
> 我们建议您为每个客户端配置一个唯一的 `RetryStrategy` 实例。如果共享 `RetryStrategy` 实例，则一个客户端出现故障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，而不是使用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”部分所述来配置此设置。

自定义可重试异常

您可以配置其他异常，以便在客户端构造期间触发重试。提供这种自定义是为了应对边缘情况，在这些情况下，引发的异常并未包含在默认的可重试异常集合中。

`retryOnException` 和 `retryOnExceptionOrCause` 方法在现有的可重试异常集合中添加新的异常类型；这不会取代默认的异常集合。通过这种方式，您可以扩展重试行为，同时维持 SDK 的默认重试功能。

如果 SDK 引发直接异常，或者异常作为“原因”封装在另一个异常中，则 `retryOnExceptionOrCause` 方法会添加一个可重试的异常。

以下代码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，或者异常已封装，则 `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();
```

> **Important**
>
> 无论重试策略配置如何，当您调用 `subscribeToShard` 方法时，都会为 Kinesis 异步客户端禁用重试。

自定义退避策略

您可以制定退避策略并将其提供给客户端。

以下代码构建了 BackoffStrategy，它会替换默认标准策略的指数延迟退避策略。

```
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（重试策略 API）将在可预见的将来得到支持。如果您当前使用 RetryPolicy 的实例来配置客户端，则系统的行为将与之前完全一致。Java SDK 在后台将其改写为 RetryStrategy。新的重试策略接口提供的功能与 RetryPolicy 相同，但创建和配置方式有所不同。

在AWS SDK for Java 2.x 中配置超时

AWS SDK for Java 2.x 提供了多层超时配置，可协助您构建弹性应用程序。SDK 提供不同类型的超时，它们协同作用来优化应用程序的性能和可靠性。

SDK 中的超时主要分为两大类：

- 服务客户端超时 - 控制 API 操作的高级别超时
- HTTP 客户端超时 - 控制网络通信的低级别超时

服务客户端超时

服务客户端超时在 API 级别运行，用于控制服务操作的整体行为，包括重试和多次尝试。

API 调用超时

API 调用超时设置整个 API 操作的最大时间，包括所有重试尝试。此超时为应用程序等待完整操作完成的时间设定了硬性限制。

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

```
    .build();
```

主要特性：

- 包括所有重试尝试。
- 包括两次重试之间等待所花费的时间。
- 提供绝对的最长等待时间。
- 防止操作无限期运行。

API 调用尝试超时

API 调用尝试超时设置单次 API 操作尝试的最长时间。如果超过此超时时间，SDK 会重试该操作（如果配置了重试），而不是使整个调用失败。

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

主要特性：

- 仅适用于单个尝试。
- 为慢速请求启用快速失败和重试。
- 必须比 API 调用超时更短。
- 有助于识别暂时性问题并从问题中恢复。

配置服务客户端超时

您可以为所有操作全局配置服务客户端超时，也可以按请求配置：

全局配置：

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

```
// When you use the s3Client for an API operation, the SDK uses the configured timeout
 values.
```

按请求配置：

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

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

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

API 超时的最佳实践

适用于 Java 的 SDK 2.x 默认不设置 API 调用超时或单个 API 调用尝试超时。为单个尝试和整个请求设置超时。当暂时性问题导致请求尝试要花费更长时间或发生致命的网络问题时，这样可以有助于您的应用程序快速失效。

HTTP 客户端超时

HTTP 客户端超时在网络级别运行，用于控制 HTTP 通信的各个方面。这些超时根据您使用的 HTTP 客户端实现而有所不同。

连接超时

连接超时用于控制与 AWS 服务 端点建立新连接时需要等待多长时间。

```
// Available with all HTTP clients.
ApacheHttpClient.builder()
```

```
    .connectionTimeout(Duration.ofSeconds(5L))
    .build();
```

目的：

- 防止因网络连接问题而挂起。
- 当服务无法访问时，会快速失效。
- 对于需要响应式错误处理的应用程序来说是必不可少的。

套接字超时（Apache 和 URLConnection 客户端）

套接字超时控制在已建立的连接上等待数据的时间。

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

读取和写入超时（Netty 客户端）

Netty 客户端为读取和写入操作提供单独的超时：

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

TLS 协商超时（Netty 客户端）

控制 TLS/SSL 握手所允许的时间：

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

连接池超时

某些 HTTP 客户端会为连接池操作提供超时：

```
ApacheHttpClient.builder()
```

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

配置 HTTP 客户端包含有关AWS SDK for Java 2.x 中的 HTTP 客户端的更多信息

超时交互和层次结构

了解不同的超时如何交互，这对于正确配置至关重要：

超时层次结构

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

配置规则

API 调用超时 ≥ API 调用尝试超时

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

API 调用尝试超时 ≥ HTTP 客户端超时

```
// HTTP client timeouts must be less than attempt timeout.
.apiCallAttemptTimeout(Duration.ofSeconds(30L))   // 30 seconds.
// HTTP client configuration.
.connectionTimeout(Duration.ofSeconds(5L))        // 5 seconds.
.readTimeout(Duration.ofSeconds(25L))             // 25 seconds (< 30).
```

考虑多次尝试

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

使用智能配置默认值

SDK 提供智能默认值，可自动配置适当的超时值：

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

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

智能默认值会自动配置：

- 连接超时值。
- TLS 协商超时值。
- 其他客户端设置。

摘要

在AWS SDK for Java 2.x 中配置有效的超时需要了解服务客户端超时和 HTTP 客户端超时之间的交互：

1. 服务客户端超时控制高级别 API 行为。
2. HTTP 客户端超时控制低级别网络行为。
3. 适当的层次结构可确保超时有效协同工作。

4. 智能默认值为大多数应用程序提供了良好的起点。

为您的使用案例适当配置超时，以便构建既能抵御网络问题又能快速响应用户的应用程序。

在AWS SDK for Java 2.x 中配置可观测性功能

遥测是自动收集和传输来自远程来源的数据，使您能够监控和分析系统的行为。

适用于 Java 的 SDK 2.x 中的可观测性功能，通过丰富的遥测数据捕获请求的完整生命周期，让开发人员可以全面了解其 Java 应用程序如何与 AWS 服务交互。这些功能使您能够收集、分析和可视化遥测信号（例如指标、日志和跟踪），让您可以监控请求模式、识别瓶颈，并优化应用程序的 AWS 交互，从而提高可靠性和性能。

主题

- 从AWS SDK for Java 2.x 发布 SDK 指标
- 监控AWS SDK for Java 2.x 应用程序
- 使用适用于 Java 的 SDK 2.x 进行日志记录

从AWS SDK for Java 2.x 发布 SDK 指标

使用AWS SDK for Java 2.x，您可以收集应用程序中服务客户端和请求的指标，在 Amazon CloudWatch Logs 中分析输出并据此采取行动。

默认情况下，SDK 中的指标收集处于禁用状态。本主题可帮助您启用和配置指标收集。

SDK 指标入门

要在应用程序中启用指标收集，请根据您的使用案例选择适当的 `MetricPublisher` 接口实现，并按照详细的设置说明进行操作：

对于长时间运行的应用程序：

- 使用 `CloudWatchMetricPublisher`
- 有关完整的设置说明、代码示例和配置选项，请参阅从长时间运行的应用程序发布 SDK 指标。

对于 AWS Lambda 函数：

- 使用 `EmfMetricLoggingPublisher`

- 有关完整的设置说明、依赖项和特定于 Lambda 的配置，请参阅为 AWS Lambda 函数发布 SDK 指标。

对于故障排除和控制台输出：

- 使用 `LoggingMetricPublisher`
- 有关本地开发和故障排除的设置说明、格式选项以及示例，请参阅将 SDK 指标输出到控制台进行开发和调试。

快速实施预览

以下是针对每个使用案例启用指标的具体示例：

长时间运行的应用程序：

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

Lambda 函数：

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

开发和调试：

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

基于 AWS CRT 的 S3 客户端的指标限制

基于 AWS CRT 的 S3 客户端目前不支持 SDK 指标收集。基于 AWS CRT 的 S3 客户端实例的生成器 `S3CrtAsyncClientBuilder` 不提供用于配置指标发布者的方法。

指标何时可用？

指标通常在 SDK for Java 发出它们后的 5-10 分钟内可用。要获得准确和最新的指标，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 请求的数量，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
- 封送、签名和 HTTP 请求等各种操作的用时
- HTTP 客户端指标，例如打开的连接数、待处理的请求数以及所使用的 HTTP 客户端的名称

> **Note**
>
> 可用指标因 HTTP 客户端而异。

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 Logs 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 Logs 指标和使用 Amazon CloudWatch Logs 警报。

使用AWS SDK for Java 2.x 从长时间运行的应用程序发布 SDK 指标

由于 `CloudWatchMetricPublisher` 实施会聚合指标并定期将指标上传到 Amazon CloudWatch，但存在延迟，因此该实施的使用非常适合长时间运行的应用程序。

指标发布者的默认设置可以更大限度地减少内存使用量和 CloudWatch 成本，同时仍能为指标数据提供一定数量的有用洞察。

设置

先完成以下步骤，然后才能使用 CloudWatchMetricPublisher 来启用和使用指标。

步骤 1：添加所需的依赖项

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。

在项目的依赖项中包括 artifactId cloudwatch-metric-publisher，并使用版本号 2.14.0 或更高版本号。

例如：

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

步骤 2：配置所需的权限

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

为特定请求启用指标

以下类演示如何为向 Amazon DynamoDB 发出的请求启用 CloudWatch 指标发布者。该类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` 实例上调用 `close`。否则可能会导致线程泄漏或文件描述符泄漏。

为特定服务客户端启用摘要指标

以下代码段演示如何为服务客户端启用采用默认设置的 CloudWatch 指标发布者。

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

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

自定义 CloudWatch 指标发布者

以下类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、指定指标发布者发送请求时所在的 AWS 区域，以及自定义发布者向 CloudWatch 发送指标的频率。

```
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();
    }
}
```

上一个代码段中显示的自定义设置具有以下效果。

- 借助 `cloudWatchClient` 方法，您可以自定义用于发送指标的 CloudWatch 客户端。在此示例中，我们使用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，其凭证将用于对发送到 CloudWatch 的请求进行身份验证。这些凭证必须具有执行 `cloudwatch:PutMetricData` 操作的权限。
- 借助 `uploadFrequency` 方法，您可以指定指标发布者将指标上传到 CloudWatch 的频率。默认为一分钟一次。
- `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，适用于 Java 的 SDK 2.x 将指标发布在命名空间 `AwsSdk/JavaSdk2` 下。您可以使用 `namespace` 方法来指定不同的值。
- 默认情况下，SDK 发布摘要指标。摘要指标包括平均值、最小值、最大值、总计和样本数量。通过在 `detailedMetrics` 方法中指定一个或多个 SDK 指标，SDK 会为每个指标发布额外数据。这些额外数据支持百分位数统计数据（例如 p90 和 p99），您可以在 CloudWatch 中进行查询。详细指标对于延迟指标（例如，用于衡量 SDK 客户端请求的端到端延迟的 `APICallDuration`）特别有用。您可以使用 `CoreMetric` 类的字段来指定其他常用 SDK 指标。

后续步骤：如果您也在使用 Lambda 函数，请参阅为 AWS Lambda 函数发布 SDK 指标，了解基于 EMF 的指标发布。

使用AWS SDK for Java 2.x 为 AWS Lambda 函数发布 SDK 指标

由于 Lambda 函数的执行时间通常为几毫秒到几分钟，而使用 `CloudWatchMetricPublisher` 发送指标存在延迟，这种延迟可能导致指标数据丢失。

EmfMetricLoggingPublisher 提供了一种更合适的方法，该方法会立即以 CloudWatch 嵌入式指标格式（EMF）将指标写成结构化日志条目。EmfMetricLoggingPublisher 适用于与 Amazon CloudWatch Logs 内置集成的执行环境，例如 AWS Lambda 和 Amazon Elastic Container Service。

设置

先完成以下步骤，然后才能使用 EmfMetricLoggingPublisher 来启用和使用指标。

步骤 1：添加所需的依赖项

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 2.30.3 或更高版本。

在项目的依赖项中包括 artifactId emf-metric-logging-publisher，并使用版本号 2.30.3 或更高版本号。

例如：

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

步骤 2：配置所需的权限

为指标发布者使用的 IAM 身份启用 logs:PutLogEvents 权限，以允许适用于 Java 的 SDK 编写 EMF 格式的日志。

步骤 3：设置日志记录

为确保正确收集指标，请将您的日志记录配置为以 INFO 级别或更低级别（例如 DEBUG）输出到控制台。在 log4j2.xml 文件中：

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

有关如何设置 log4j2.xml 文件的更多信息，请参阅本指南中的日志记录主题。

配置和使用 **EmfMetricLoggingPublisher**。

以下 Lambda 函数类先创建和配置 EmfMetricLoggingPublisher 实例，然后将其与 Amazon DynamoDB 服务客户端一起使用：

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

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

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

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

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

        dynamoDb.putItem(putItemRequest);

        return "Request handled";
    }
}
```

当 DynamoDB 客户端执行 `putItem` 方法时，它会以 EMF 格式自动将指标发布到 CloudWatch 日志流。

EMF 日志事件的示例

例如，如果您向 GameHandler Lambda 函数发送以下事件，并且之前日志记录按如下所示进行配置：

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

函数处理事件后，您会看到两个与以下示例类似的日志事件。第二个事件中的 JSON 对象包含 DynamoDB `PutItem` 操作的 Java SDK 指标数据。

当 CloudWatch 收到 EMF 格式的日志事件时，它会自动解析结构化的 JSON 以提取指标数据。然后，CloudWatch 会创建相应的指标，同时将原始日志条目存储在 CloudWatch Logs 中。

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

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

```
                    [
                        "OperationName",
                        "ServiceId"
                    ]
                ],
                "Metrics": [
                    {
                        "Name": "AvailableConcurrency"
                    },
                    {
                        "Name": "PendingConcurrencyAcquires"
                    },
                    {
                        "Name": "ServiceCallDuration",
                        "Unit": "Milliseconds"
                    },
                    {
                        "Name": "EndpointResolveDuration",
                        "Unit": "Milliseconds"
                    },
                    {
                        "Name": "MaxConcurrency"
                    },
                    {
                        "Name": "BackoffDelayDuration",
                        "Unit": "Milliseconds"
                    },
                    {
                        "Name": "MarshallingDuration",
                        "Unit": "Milliseconds"
                    },
                    {
                        "Name": "LeasedConcurrency"
                    },
                    {
                        "Name": "SigningDuration",
                        "Unit": "Milliseconds"
                    },
                    {
                        "Name": "ConcurrencyAcquireDuration",
                        "Unit": "Milliseconds"
                    },
                    {
                        "Name": "ApiCallSuccessful"
```

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

`EmfMetricLoggingPublisher.Builder` 的 API 文档显示了您可以使用的配置选项。

您还可以为单个请求启用 EMF 指标日志记录，如 CloudWatchMetricPublisher 所示。

后续步骤：对于长时间运行的应用程序，请参阅从长时间运行的应用程序发布 SDK 指标，了解基于 CloudWatch 的指标发布。

使用AWS SDK for Java 2.x 向控制台输出 SDK 指标

`LoggingMetricPublisher` 实现将指标直接输出到应用程序的控制台或日志文件。这种方法非常适合开发、调试和了解 SDK 收集的指标，而无需像 Amazon CloudWatch 这样的外部服务。

与 `CloudWatchMetricPublisher` 和 `EmfMetricLoggingPublisher` 不同，`LoggingMetricPublisher` 提供即时输出，没有延迟或外部依赖项。这使其非常适合用于本地开发和故障排除场景。

何时使用 LoggingMetricPublisher

在需要执行以下操作时使用 `LoggingMetricPublisher`：

- 在开发过程中调试指标收集
- 了解 SDK 为您的运营收集了哪些指标
- 在本地排查性能问题
- 在没有外部服务依赖项的情况下测试指标收集
- 立即在控制台或日志文件中查看指标

> **Note**
>
> 不建议将 `LoggingMetricPublisher` 用于需要持久指标存储和分析功能的生产环境中。

为指标设置控制台日志记录

要查看 `LoggingMetricPublisher` 输出，请将您的日志记录框架配置为显示 `INFO` 级别消息。以下 `log4j2.xml` 配置可确保指标显示在控制台中：

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="WARN">
    <Appenders>
        <Console name="ConsoleAppender" target="SYSTEM_OUT">
            <PatternLayout pattern="%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg
%n"/>
        </Console>
```

```
    </Appenders>
    <Loggers>
        <Root level="INFO">
            <AppenderRef ref="ConsoleAppender"/>
        </Root>
        <!-- Ensure LoggingMetricPublisher output appears. -->
        <Logger name="software.amazon.awssdk.metrics.LoggingMetricPublisher"
 level="INFO" />
    </Loggers>
</Configuration>
```

此配置指示 SDK 在 INFO 级别将指标输出到控制台。LoggingMetricPublisher 记录器配置可确保即使根记录器使用更高的级别（如 WARN 或 ERROR），也能显示指标输出。

为服务客户端启用控制台指标

以下示例演示如何创建 LoggingMetricPublisher 并与 Amazon Simple Storage Service 客户端一起使用：

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

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

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

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

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

选择指标输出格式

LoggingMetricPublisher 支持两种输出格式：

- PLAIN 格式（默认）：以紧凑的单行形式输出指标
- PRETTY 格式：以多行、用户可读的格式输出指标

以下示例演示如何使用 PRETTY 格式，以便在开发过程中更轻松地读取输出：

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

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

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

完整示例

以下示例演示了通过两种方式来使用 LoggingMetricPublisher：

- 在服务客户端级别
- 对于单个请求

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

/**
 * Demonstrates how to use LoggingMetricPublisher with AWS S3 SDK for Java 2.x.
 * <p>
 * This demo focuses on the S3 listBuckets operation to show how metrics are collected
```

```
 * and logged to the console for development and debugging purposes.
 * <p>
 * LoggingMetricPublisher is ideal for:
 * - Development and debugging
 * - Console output for troubleshooting
 * - Understanding what metrics are being collected
 * - Testing metric collection without external dependencies
 */
public class S3LoggingMetricPublisherDemo {

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

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

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

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

        MetricPublisher metricPublisher = LoggingMetricPublisher.create();

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

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

            // The next block shows the using a different LoggingMetricPublisher with a
 `PRETTY` format.
```

```
            // Since the metric publisher is added to the request using the
 `overrideConfiguration`, this formatting
            // applies only to the one request.
            try {
                s3Client.listBuckets(ListBucketsRequest.builder()
                        .overrideConfiguration(config -> config
                                .addMetricPublisher(LoggingMetricPublisher.create(
                                        Level.INFO,
 LoggingMetricPublisher.Format.PRETTY)))
                        .build());
            } catch (Exception e) {
                logger.info("Request failed with metrics logged: {}", e.getMessage());
            }
            logger.info("Found {} buckets in your AWS account.",
 response.buckets().size());

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

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

该代码将以下内容记录到控制台：

```
INFO  LoggingMetricPublisher - Metrics published: MetricCollection(name=ApiCall,
 metrics=[MetricRecord(metric=MarshallingDuration, value=PT0.005409792S),
 MetricRecord(metric=RetryCount, value=0), MetricRecord(metric=ApiCallSuccessful,
 value=true), MetricRecord(metric=OperationName, value=ListBuckets),
 MetricRecord(metric=EndpointResolveDuration, value=PT0.000068S),
 MetricRecord(metric=ApiCallDuration, value=PT0.163802958S),
 MetricRecord(metric=CredentialsFetchDuration, value=PT0.145686542S),
 MetricRecord(metric=ServiceEndpoint, value=https://s3.amazonaws.com),
 MetricRecord(metric=ServiceId, value=S3)],
 children=[MetricCollection(name=ApiCallAttempt,
 metrics=[MetricRecord(metric=TimeToFirstByte, value=PT0.138816S),
 MetricRecord(metric=SigningDuration, value=PT0.007803459S),
 MetricRecord(metric=ReadThroughput, value=165153.96002660287),
 MetricRecord(metric=ServiceCallDuration, value=PT0.138816S),
```

```
 MetricRecord(metric=AwsExtendedRequestId, value=e13Swj3uwn0qP1Oz
+m7II5OGq7jf8xxT8H18iDfRBCQmDg+gU4ek91Xrsl8XxRLROlIzCAPQtsQF0DAAWOb8ntuKCzX2AJdj),
 MetricRecord(metric=HttpStatusCode, value=200),
 MetricRecord(metric=BackoffDelayDuration, value=PT0S),
 MetricRecord(metric=TimeToLastByte, value=PT0.148915667S),
 MetricRecord(metric=AwsRequestId, value=78AW9BM7SWR6YMGB)],
 children=[MetricCollection(name=HttpClient,
 metrics=[MetricRecord(metric=MaxConcurrency, value=50),
 MetricRecord(metric=AvailableConcurrency, value=0),
 MetricRecord(metric=LeasedConcurrency, value=1),
 MetricRecord(metric=ConcurrencyAcquireDuration, value=PT0.002623S),
 MetricRecord(metric=PendingConcurrencyAcquires, value=0),
 MetricRecord(metric=HttpClientName, value=Apache)], children=[])])])
INFO  LoggingMetricPublisher - [4e6f2bb5] ApiCall
INFO  LoggingMetricPublisher - [4e6f2bb5] ############################################
INFO  LoggingMetricPublisher - [4e6f2bb5] # MarshallingDuration=PT0.000063S          #
INFO  LoggingMetricPublisher - [4e6f2bb5] # RetryCount=0                            #
INFO  LoggingMetricPublisher - [4e6f2bb5] # ApiCallSuccessful=true                  #
INFO  LoggingMetricPublisher - [4e6f2bb5] # OperationName=ListBuckets               #
INFO  LoggingMetricPublisher - [4e6f2bb5] # EndpointResolveDuration=PT0.000024375S   #
INFO  LoggingMetricPublisher - [4e6f2bb5] # ApiCallDuration=PT0.018463083S           #
INFO  LoggingMetricPublisher - [4e6f2bb5] # CredentialsFetchDuration=PT0.000022334S  #
INFO  LoggingMetricPublisher - [4e6f2bb5] # ServiceEndpoint=https://s3.amazonaws.com #
INFO  LoggingMetricPublisher - [4e6f2bb5] # ServiceId=S3                            #
INFO  LoggingMetricPublisher - [4e6f2bb5] ############################################
INFO  LoggingMetricPublisher - [4e6f2bb5]     ApiCallAttempt
INFO  LoggingMetricPublisher - [4e6f2bb5]
 ##########################################################################################
INFO  LoggingMetricPublisher - [4e6f2bb5]     # TimeToFirstByte=PT0.0165575S
                                                                          #
INFO  LoggingMetricPublisher - [4e6f2bb5]     # SigningDuration=PT0.000301125S
                                                                          #
INFO  LoggingMetricPublisher - [4e6f2bb5]     # ReadThroughput=1195591.792850103
                                                                          #
INFO  LoggingMetricPublisher - [4e6f2bb5]     # ServiceCallDuration=PT0.0165575S
                                                                          #
INFO  LoggingMetricPublisher - [4e6f2bb5]     #
 AwsExtendedRequestId=3QI1eenRuokdszWqZBmBMDUmko6FlSmHkM
+CUMNMeLor7gJml4D4lv6QXUZ1zWoTgG+tHbr6yo2vHdz4h1P8PDovvtMFRCeB #
INFO  LoggingMetricPublisher - [4e6f2bb5]     # HttpStatusCode=200
                                                                          #
INFO  LoggingMetricPublisher - [4e6f2bb5]     # BackoffDelayDuration=PT0S
                                                                          #
```

```
INFO  LoggingMetricPublisher - [4e6f2bb5]     # TimeToLastByte=PT0.017952625S
                                                                              #
INFO  LoggingMetricPublisher - [4e6f2bb5]     # AwsRequestId=78AVFAF795AAWAXH
                                                                              #
INFO  LoggingMetricPublisher - [4e6f2bb5]
 ################################################################################################
INFO  LoggingMetricPublisher - [4e6f2bb5]         HttpClient
INFO  LoggingMetricPublisher - [4e6f2bb5]
 ########################################
INFO  LoggingMetricPublisher - [4e6f2bb5]         # MaxConcurrency=50
   #
INFO  LoggingMetricPublisher - [4e6f2bb5]         # AvailableConcurrency=0
   #
INFO  LoggingMetricPublisher - [4e6f2bb5]         # LeasedConcurrency=1
   #
INFO  LoggingMetricPublisher - [4e6f2bb5]         #
 ConcurrencyAcquireDuration=PT0.00004S #
INFO  LoggingMetricPublisher - [4e6f2bb5]         # PendingConcurrencyAcquires=0
   #
INFO  LoggingMetricPublisher - [4e6f2bb5]         # HttpClientName=Apache
   #
INFO  LoggingMetricPublisher - [4e6f2bb5]
 ########################################
INFO  S3LoggingMetricPublisherDemo - Found 6 buckets in your AWS account.
```

该示例的其他构件

Maven pom.xml 文件

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

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

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

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

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

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

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

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

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

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

Log4j2.xml 配置文件

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

这些指标包括计时信息、服务详细信息、操作名称和 HTTP 状态代码，可帮助您了解应用程序的 AWS API 使用模式。

后续步骤

在使用 `LoggingMetricPublisher` 进行开发和调试之后，请考虑在生产环境中使用以下选项：

- 对于长时间运行的应用程序，使用 CloudWatchMetricPublisher 将指标发送到 Amazon CloudWatch 进行分析和提醒
- 对于 AWS Lambda 函数，使用 EmfMetricLoggingPublisher，以 CloudWatch 嵌入式指标格式发布指标

AWS SDK for Java 2.x：全面的指标参考

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 SDK 启用和配置指标的更多信息，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	API 调用的持续时间。这包括所有已发起的调用尝试。	持续时间*
ApiCallSuccessful	如果 API 调用成功则为 true，否则为 false。	布尔值
CredentialsFetchDuration	获取 API 调用的签名凭证的持续时间。	持续时间*
EndpointResolveDuration	解析 API 调用所用端点的持续时间。	持续时间*
MarshallingDuration	将 SDK 请求封送到 HTTP 请求的持续时间。	持续时间*

指标名称	描述	类型
OperationName	所调用服务操作的名称。	字符串
RetryCount	SDK 在执行请求时自动重试的次数。0 表示该请求第一次就成功了，没有进行任何重试。 有关配置重试行为的更多信息，请参阅重试策略。	整数
ServiceId	服务的唯一 ID。	字符串
ServiceEndpoint	服务的端点。	URI
TokenFetchDuration	获取 API 调用的签名凭证的持续时间。	持续时间*

*java.time.Duration。

为每次请求尝试收集的指标

每个 API 调用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。	字符串
AwsRequestId	服务请求的请求 ID。	字符串
BackoffDelayDuration	在这次 API 调用尝试之前 SDK 已等待的持续时间。该值基于客户端上 `BackoffStrategy` 设置。有关更多信息，请参阅本指南的重试策略部分。	持续时间*

指标名称	描述	类型
ErrorType	进行调用尝试时发生的错误类型。 以下是可能的值： • `Throttling` ：服务响应时出现节流错误。 • `ServerError` ：服务响应时出现节流以外的错误。 • `ConfiguredTimeout` ：在 API 调用级别或 API 调用尝试级别出现客户端超时。 • `IO`：发生了 I/O 错误。 • `Other`：用于捕获所有未归入上述类别中的其他错误的“兜底”分类。	字符串
ReadThroughput	客户端的读取吞吐量，定义为 `NumberOfResponseBytesRead / (TTLB - TTFB)`。此值以每秒字节数为单位。 请注意，此指标仅测量从 `ResponseTransformer` 或 `AsyncResponseTransformer` 内读取的字节。计算中不包括在转换器外部读取的数据（例如，当响应流作为转换器的结果返回时）。	双精度

指标名称	描述	类型
ServiceCallDuration	连接到服务（或从连接池中获取连接）、发送序列化请求和接收初始响应（例如 HTTP 状态码和标头）的持续时间。这不包括从服务中读取整个响应的时间。	持续时间*
SigningDuration	签署 HTTP 请求的持续时间。	持续时间*
TimeToFirstByte	从将 HTTP 请求发送到服务（包括获取连接），到接收响应中标头的第一个字节的持续时间。	持续时间*
TimeToLastByte	从将 HTTP 请求发送到服务（包括获取连接），到接收响应的最后一个字节的持续时间。 请注意，对于返回流式响应的 API，此指标涵盖了从开始直至 `ResponseTransformer` 或 `AsyncResponseTransformer` 完成的时间。	持续时间*
UnmarshallingDuration	将 HTTP 响应解组到 SDK 响应的持续时间。 注意：对于流式操作，这不包括读取响应有效载荷的时间。	持续时间*

*java.time.Duration。

HTTP 指标

指标名称	描述	类型	需要 HTTP 客户端*
AvailableConcurrency	在不与目标服务器建立新连接的情况下，HTTP 客户端支持的额外并发请求数。 对于 HTTP/1 操作，这等于与服务建立的空闲 TCP 连接的数量。对于 HTTP/2 操作，这等于空闲流的数量。 注意：此值因 HTTP 客户端实现而异： • Apache 客户端：值适用于整个 HTTP 客户端 • Netty 客户端：对每个端点单独计算值 • 基于 AWS CRT 的客户端：对每个端点单独计算值 该值的范围限定于单个 HTTP 客户端实例，不包括同一 JVM 中其他 HTTP 客户端的并发。	整数	Apache、Netty、CRT
ConcurrencyAcquireDuration	从连接池中获取通道的持续时间。 对于 HTTP/1 操作，通道等于 TCP 连接。对于 HTTP/2 操作，通道等于 HTTP/2 流通道。 获取新通道可能包括以下时间： 1. 正在等待并发许可，受客户端最大并发配置的限制。 2. 如果池中没有可用的现有连接，则建立新连接。 3. 如果启用了 TLS，则执行 TLS 握手和协商。	持续时间*	Apache、Netty、CRT
HttpClientName	用于请求的 HTTP 的名称。	字符串	Apache、Netty、CRT
HttpStatusCode	HTTP 响应的状态代码。	整数	任何

指标名称	描述	类型	需要 HTTP 客户端*
LeasedConcurrency	HTTP 客户端当前执行的请求数。 对于 HTTP/1 操作，这等于与服务建立的活跃 TCP 连接的数量（不包括空闲连接）。对于 HTTP/2 操作，这等于服务的活动 HTTP 流的数量（不包括空闲流容量）。 注意：此值因 HTTP 客户端实现而异： • Apache 客户端：值适用于整个 HTTP 客户端 • Netty 客户端：对每个端点单独计算值 • 基于 AWS CRT 的客户端：对每个端点单独计算值 该值的范围限定于单个 HTTP 客户端实例，不包括同一 JVM 中其他 HTTP 客户端的并发。	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。 对于 HTTP/1 操作，这等于 HTTP 客户端可以汇集的最大 TCP 连接数。对于 HTTP/2 操作，这等于 HTTP 客户端可以汇集的最大流数。 注意：此值因 HTTP 客户端实现而异： • Apache 客户端：值适用于整个 HTTP 客户端 • Netty 客户端：对每个端点单独计算值 • 基于 AWS CRT 的客户端：对每个端点单独计算值 该值的范围限定于单个 HTTP 客户端实例，不包括同一 JVM 中其他 HTTP 客户端的并发。	整数	Apache、Netty、CRT

指标名称	描述	类型	需要 HTTP 客户端*
PendingConcurrencyAcquires	等待 HTTP 客户端并发的请求数。 对于 HTTP/1 操作，这等于等待 TCP 连接建立或从连接池返回的请求数。对于 HTTP/2 操作，这等于等待来自连接池的新流（可能还有新 HTTP/2 连接）的请求数。 注意：此值因 HTTP 客户端实现而异： • Apache 客户端：值适用于整个 HTTP 客户端 • Netty 客户端：对每个端点单独计算值 • 基于 AWS CRT 的客户端：对每个端点单独计算值 该值的范围限定于单个 HTTP 客户端实例，不包括同一 JVM 中其他 HTTP 客户端的并发。	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。

列中所用术语解释：

- Apache：基于 Apache 的 HTTP 客户端（`ApacheHttpClient`）
- Netty：基于 Netty 的 HTTP 客户端（`NettyNioAsyncHttpClient`）
- CRT：基于 AWS CRT 的 HTTP 客户端（`AwsCrtAsyncHttpClient`）
- 任意：指标数据的收集不依赖于 HTTP 客户端；这包括基于 URLConnection 的 HTTP 客户端（`UrlConnectionHttpClient`）

监控AWS SDK for Java 2.x 应用程序

使用AWS SDK for Java 2.x 时为了使应用程序保持可靠性、可用性和性能，监控是一个很重要的部分。AWS 提供了以下监控工具来监控适用于 Java 的 SDK 2.x、在出现错误时进行报告并适时自动采取措施：

- Amazon CloudWatch 可实时监控您的 AWS 资源以及您在 AWS 上运行的应用程序。您可以收集和跟踪指标，创建自定义的控制平面，以及设置警报以在指定的指标达到您指定的阈值时通知您或采取措施。例如，您可以使用 CloudWatch 跟踪 Amazon EC2 实例的 CPU 使用率或其他指标并且在需要时自动启动新实例。有关更多信息，请参阅《Amazon CloudWatch 用户指南》https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/。
- Amazon CloudWatch Logs 使您能够监控、存储和访问来自 Amazon EC2 实例、CloudTrail 和其他来源的日志文件。CloudWatch Logs 可以监控日志文件中的信息，并在达到特定阈值时通知您。您还可以在高持久性存储中检索您的日志数据。有关更多信息，请参阅 Amazon CloudWatch Logs 用户指南。
- AWS CloudTrail 捕获由您的 AWS 账户或代表该账户发出的 API 调用和相关事件，并将日志文件传输到您指定的 Amazon S3 存储桶。您可以标识哪些用户和账户调用了 AWS、发出调用的源 IP 地址以及调用的发生时间。有关更多信息，请参阅《AWS CloudTrail 用户指南》。

使用适用于 Java 的 SDK 2.x 进行日志记录

AWS SDK for Java 2.x 通过 SLF4J，它是一个抽象层，支持在运行时使用多种日志记录系统中的一个。

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AWS SDK for Java 2.x 结合使用的主要日志层次结构是 `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 AWS 服务的所有请求都会生成一个唯一的 AWS 请求 ID，如果您遇到与 AWS 服务如何处理请求有关的问题，可以使用它。如果调用任何服务时失败，可以通过 SDK 中的 SdkServiceException 对象来编程访问 AWS 请求 ID，该 ID 还可以在“software.amazon.awssdk.request”记录器中通过“DEBUG”日志级别进行报告。

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

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

调试级别 SDK 日志记录

如果您需要更详细地了解 SDK 正在做什么，可以将 `software.amazon.awssdk` 记录器的日志记录级别设置为 DEBUG。在此级别，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。

在此日志记录级别，SDK 会记录有关配置、凭证解析、执行拦截器、高级 TLS 活动、请求签名等方面的信息。

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

```
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>
```

启用线路日志记录

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临时启用它。

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 S3AsyncClient）使用 Netty 非阻塞 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
AwsCrtHttpClient	

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，详细线路日志记录也可能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` 文件会为 Apache HttpClient 启用完整线路日志记录。

```
<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: sAFeZDOKdUMsBbkdjyDZw7P0oocb4C9KbiuzfJ6TWKQsGXHM/
dFuOvr2tUb7Y1wEHGdJ3DSIxq0=}{x-amz-request-id: P9QW9SMZ97FKZ9X7}{Date: Tue,
 27 Sep 2022 13:39:57 GMT}{Last-Modified: Tue, 13 Sep 2022 14:38:12 GMT}{ETag:
 "2cbe5ad4a064cedec33b452bebf48032"}{x-amz-transfer-encoding: append-md5}{Accept-
Ranges: bytes}{Content-Type: text/plain}{Server: AmazonS3}{Content-Length: 67}
```

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/AwsCrtHttpClient

如果您已将服务客户端配置为使用基于 AWS CRT 的 HTTP 客户端的实例，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，基于 AWS CRT 的 HTTP 客户端仅记录响应标头。不记录请求标头、请求正文和响应正文。

在AWS SDK for Java 2.x 中配置客户端端点

适用于 Java 的 SDK 2.x 提供了多种配置服务端点的方法。端点是 SDK 用来向 AWS 服务进行 API 调用的 URL。默认情况下，SDK 会根据您配置的 AWS 区域自动为每项服务确定适合的端点。但在某些情况下，您可能需要自定义或覆盖这些端点：

- 使用本地或第三方服务实现（例如 LocalStack）
- 通过代理或 VPC 端点连接到 AWS 服务
- 针对测试版或预发布版服务端点进行测试

端点配置选项

AWS SDK for Java 2.x 提供了几种配置端点的方法：

- 使用服务客户端生成器进行代码内配置

- 使用环境变量进行外部配置
- 使用 JVM 系统属性进行外部配置
- 使用共享 AWS 配置文件进行外部配置

代码内端点配置

使用 **endpointOverride**

配置端点的最直接方法是在服务客户端生成器中使用 endpointOverride 方法。此方法接受表示自定义端点 URL 的 URI 对象。

Example为 Amazon S3 客户端设置自定义端点

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

S3Client s3 = S3Client.builder()
        .region(Region.US_WEST_2)
        .endpointOverride(URI.create("https://my-custom-s3-endpoint.example.com"))
        .build();
```

使用 endpointOverride 时，即使已明确设置端点，您仍必须为客户端指定区域。区域用于对请求进行签名。

端点发现

有些 AWS 服务支持端点发现，SDK 可以自动发现要使用的最佳端点。您可以在服务客户端生成器中使用 endpointDiscoveryEnabled 方法来启用或禁用此功能。

Example为 DynamoDB 客户端启用端点发现

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

DynamoDbClient dynamoDb = DynamoDbClient.builder()
        .region(Region.US_WEST_2)
```

```
        .endpointDiscoveryEnabled(true)
        .build();
```

请求级别端点配置

在某些情况下，您可能需要覆盖特定请求的端点，而为其他具有默认端点的请求使用相同的客户端。AWS SDK for Java 2.x 通过请求覆盖来支持此功能。

Example覆盖特定请求的端点

```
import software.amazon.awssdk.core.SdkRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.http.SdkHttpRequest;

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

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

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

请注意，请求级别的端点覆盖受到限制，可能并不适用于所有服务或场景。在大多数情况下，建议使用客户端级别端点配置。

外部端点配置

使用环境变量

您可以使用环境变量来配置端点。SDK 支持通过 `AWS_ENDPOINT_URL_[SERVICE]` 格式的环境变量来配置服务特定的端点，其中 `[SERVICE]` 是大写服务标识符。

Example使用环境变量设置 S3 端点

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

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

您还可以使用以下环境变量设置全局端点 URL 前缀或后缀：

- `AWS_ENDPOINT_URL` - 为所有服务设置全局端点
- `AWS_ENDPOINT_URL_PREFIX` - 为所有服务端点添加前缀
- `AWS_ENDPOINT_URL_SUFFIX` - 为所有服务端点添加后缀

使用 JVM 系统属性

您也可以使用 JVM 系统属性来配置端点。其格式与环境变量类似，但使用不同的命名约定。

Example使用 JVM 系统属性设置 S3 端点

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

也可以通过系统属性进行全局端点配置：

- `aws.endpointUrl` - 为所有服务设置全局端点
- `aws.endpointUrl.prefix` - 为所有服务端点添加前缀
- `aws.endpointUrl.suffix` - 为所有服务端点添加后缀

使用共享 AWS 配置文件

AWS SDK for Java 2.x 还支持通过共享 AWS 配置文件进行端点配置，该文件通常位于 `~/.aws/config`（Linux/macOS）或 `%USERPROFILE%\.aws\config`（Windows）。有关信息和示例，请参阅《AWS SDK 和工具参考指南》。

配置优先级

当存在多个端点配置时，SDK 遵循以下优先顺序（从最高到最低）：

1. 请求级别覆盖（如果适用）
2. 通过 `endpointOverride` 进行客户端级别配置
3. 环境变量
4. JVM 系统属性
5. 共享 AWS 配置文件
6. 基于已配置 AWS 区域的默认端点

特定于服务的端点配置

有些 AWS 服务还具有特定于该服务的额外端点配置选项。下面是几个示例：

Amazon S3 端点配置

Amazon S3 通过 `S3Configuration` 类支持几种端点配置：

- `dualstackEnabled` - 启用 IPv6 支持
- `accelerateModeEnabled` - 启用 S3 Transfer Acceleration
- `pathStyleAccessEnabled` - 使用路径风格访问方式而不是虚拟托管风格
- `useArnRegionEnabled` - 为跨区域请求使用 ARN 中的区域
- `fipsModeEnabled` - 将请求路由到符合 FIPS 标准的端点

Example配置特定于 S3 的端点选项

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

S3Client s3 = S3Client.builder()
        .region(Region.US_WEST_2)
        .serviceConfiguration(S3Configuration.builder()
                .accelerateModeEnabled(true)
                .dualstackEnabled(true)
```

```
                .pathStyleAccessEnabled(false)
                .fipsModeEnabled(true)
                .build())
        .build();
```

DynamoDB 端点配置

对于 DynamoDB，您可能需要使用端点发现或连接到 DynamoDB local 进行测试：

Example连接到 DynamoDB local

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

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

DynamoDB 还支持使用基于账户的端点，您可以在代码中配置或使用外部设置进行配置。以下示例说明如何在创建客户端时在代码中禁用基于账户的端点（默认设置为首选）：

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

最佳实践

在AWS SDK for Java 2.x 中配置端点时，请考虑以下最佳实践：

- 为环境特定的端点使用外部配置 - 使用环境变量、系统属性或 AWS 配置文件来配置不同环境（开发、测试、生产）的端点。
- 对特定于应用程序的端点使用代码内配置 - 使用客户端生成器的 `endpointOverride` 方法来处理特定于应用程序设计的端点。

- 始终指定区域 - 即使在覆盖端点时，也始终指定用于请求签名的区域。
- 请谨慎使用全局端点覆盖 - 使用全局端点覆盖会影响所有服务，而这可能不是您想要的结果。
- 考虑安全影响 - 使用自定义端点时，请确保为这些端点采取了适当的安全措施，尤其是对于生产工作负载。

在中配置 HTTP 客户端 AWS SDK for Java 2.x

您可以使用 AWS SDK for Java 2.x更改用于服务客户端的 HTTP 客户端，也可以更改 HTTP 客户端的默认配置。本部分讨论 SDK 的 HTTP 客户端和设置。

SDK for Java 中可用的 HTTP 客户端

同步客户端

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

ApacheHttpClient（默认）

ApacheHttpClient是同步服务客户端的默认 HTTP 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 “配置基于 AWS CRT 的 HTTP 客户端”。

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。

Apache5 HttpClient

Apache5 HttpClient 是在 Apache 5.x 上构建*`ApacheHttpClient`*的更新版本。 HttpClient 版本 5.x 是 Apache 积极维护的版本，`ApacheHttpClient`通过引入现代 Java 生态系统兼容性（包括对 Java 21 的虚拟线程支持）以及通过 SLF4 J 增强的日志灵活性，对之

前的版本进行了改进，Apache5HttpClient将在未来版本的 Java AWS SDK 2.x 中取代ApacheHttpClient默认同步客户端。它具有相同的 API 和功能，非常适合作为即用型替代品。有关配置 Apache5HttpClient 的信息，请参阅配置基于 Apache 5.x 的 HTTP 客户端。

异步客户端

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

NettyNioAsyncHttpClient （默认）

NettyNioAsyncHttpClient是异步客户端使用的默认 HTTP 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

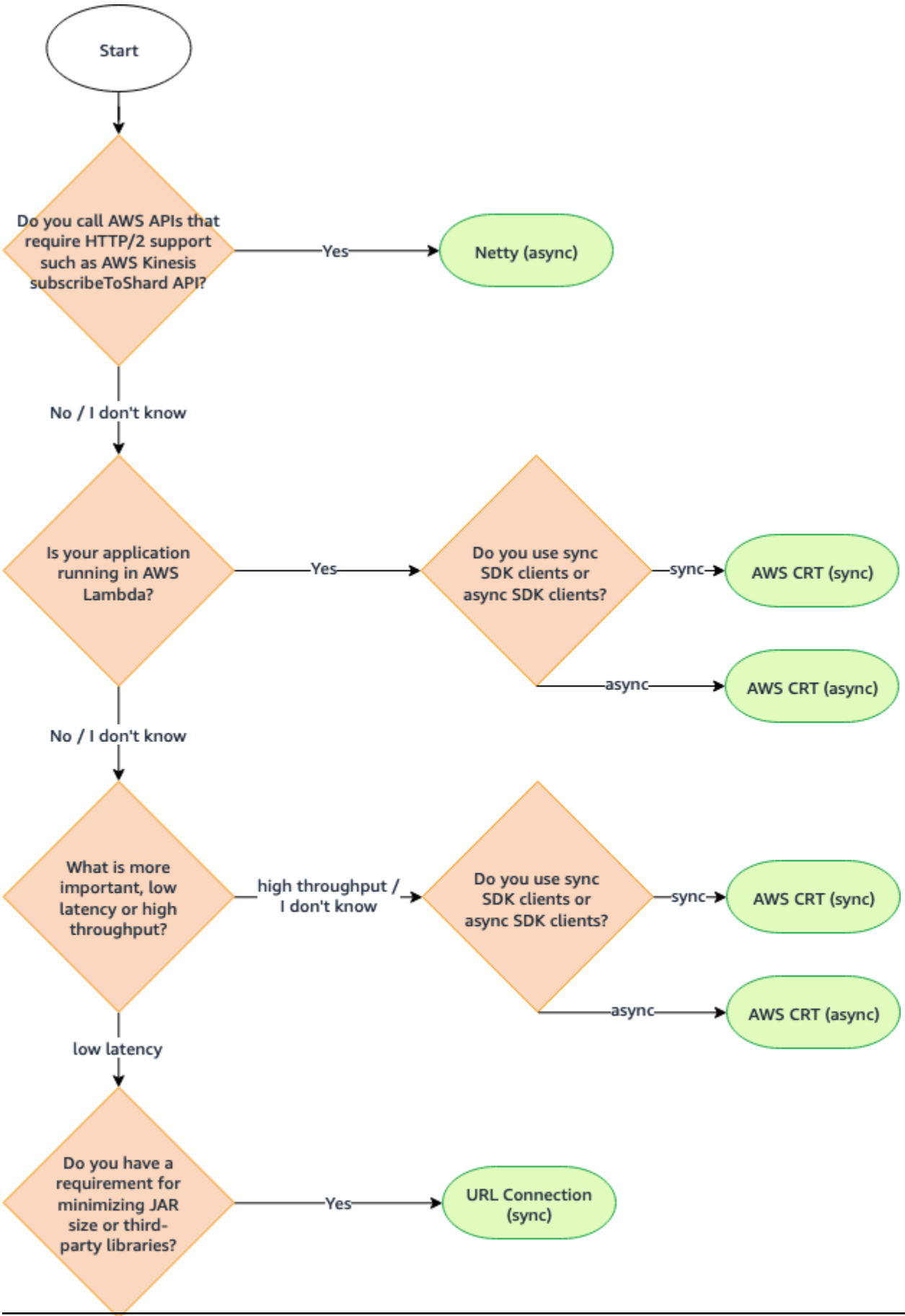
基于 AWS 通用运行时 (CRT) HTTP 客户端。AwsCrtAsyncHttpClient有关配置 AwsCrtAsyncHttpClient 的信息，请参阅the section called “配置基于 AWS CRT 的 HTTP 客户端”。

HTTP 客户端建议

选择 HTTP 客户端实现时，应考虑几个因素。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，有助于确定使用哪个 HTTP 客户端。


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

HTTP 客户端比较

下表提供了各 HTTP 客户端的详细信息。

HTTP 客户端	同步还是异步	何时使用	限制/缺点
基于 Apache 的 HTTP 客户端 （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，启动较慢
基于 URLConnection 的 HTTP 客户端	同步	如果您对限制第三方依赖项有硬性要求，请使用它	不支持某些 API 所要求的 HTTP 补丁方法，例如亚马逊 APIGateway 更新操作
AWS 基于 CRT 同步的 HTTP 客户端 [1]	同步	• 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： • javax.net.ssl.keyStore • javax.net.ssl。keyStorePassword • javax.net.ssl.trustStore • javax.net.ssl。 trustStorePassword
基于 Netty 的 HTTP 客户端 （默认异步 HTTP 客户端）	异步	• 如果您的应用程序调用需要 HTTP/2 支持（例如 Kinesis API） APIs ，请使用它 SubscribeToShard	与其他 HTTP 客户端相比，启动较慢

HTTP 客户端	同步还是异步	何时使用	限制/缺点
AWS 基于 CRT 的异步 HTTP 客户端 [1]	异步	• 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	• 不支持需要 HTTP/2 支持的服务客户端，例如 `KinesisAsynClient` 和 `TranscribeStreamingAsyncClient` 不支持以下 Java 系统属性： • javax.net.ssl.keyStore • javax.net.ssl。keyStorePassword • javax.net.ssl.trustStore • javax.net.ssl。 trustStorePassword

[1] 由于基于 CRT 的 HTTP 客户端，我们建议您尽可能使用 AWS 基于 CRT 的 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 HTTP 客户端的两个属性以及不影响 HTTP 客户端的其他属性。

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	`export AWS_DEFAULTS_MODE=auto`
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 配置文件在全局范围内设置了默认模式，则在生成 HTTP 客户端时可以覆盖这些设置。

使用 httpClientBuilder() 方法生成 HTTP 客户端时，设置仅适用于您正在生成的实例。此处显示了此方法一个示例。本示例中基于 Netty 的 HTTP 客户端会覆盖在全局范围内为 connectTimeoutInMillis 和 tlsNegotiationTimeoutInMillis 设置的任何默认模式值。

配置基于 Apache 的 HTTP 客户端

AWS SDK for Java 2.x 中的同步服务客户端使用基于 Apache 的 HTTP 客户端，默认为 ApacheHttpClient。SDK 的 ApacheHttpClient 基于 Apache HttpClient。

SDK 还提供了 UrlConnectionHttpClient，它加载速度更快，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 ApacheHttpClient 配置选项，请参阅 ApacheHttpClient.Builder 和 ProxyConfiguration.Builder。

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>2.27.21</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
<dependencies>
    <!-- The s3 dependency automatically adds a runtime dependency on the
 ApacheHttpClient-->
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>s3</artifactId>
    </dependency>
</dependencies>
```

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>2.27.21</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
```

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

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和内存使用量，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

与默认的 `ApacheHttpClient` 相比，AWS SDK for Java 2.x 提供了更轻量的 `UrlConnectionHttpClient` HTTP 客户端。`UrlConnectionHttpClient` 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 PATCH 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 CSPM API 中的几个 `Update*` 操作以及 BatchUpdateFindings 操作
> - 所有 Amazon API Gateway API 的 `Update*` 操作
>
> 如果您可能使用 `UrlConnectionHttpClient`，请先参阅您正在使用的 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>2.27.21</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>

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

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

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。

无论哪种方法，您都需要使用 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和内存使用量，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 客户端当前不支持 HTTPS 代理系统属性或 HTTPS_PROXY 环境变量。

配置基于 Netty 的 HTTP 客户端

AWS SDK for Java 2.x 中用于异步操作的默认 HTTP 客户端是基于 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>2.27.21</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
<dependencies>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>dynamodb-enhanced</artifactId>
    </dependency>
</dependencies>
```

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>2.27.21</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>dynamodb-enhanced</artifactId>
        </dependency>
        <!-- By adding the netty-nio-client dependency, NettyNioAsyncHttpClient will
 be
             added to the compile classpath so you can configure it. -->
        <dependency>
```

```
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>netty-nio-client</artifactId>
        </dependency>
    </dependencies>
```

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和内存使用量，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.
```

配置 ALPN 协议协商

ALPN（应用层协议协商）是一个 TLS 扩展，该扩展让应用层可以协商在建立安全连接时应执行哪个协议，从而避免额外的网络往返并提供更好的性能。

要使基于 Netty 的 HTTP 客户端能够使用 ALPN，请调用生成器方法，如以下代码段所示：

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


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

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

 .httpClient(nettyClient)

 .build();
```

如前面的代码段所示，ALPN 协议协商目前仅适用于 HTTP/2 协议。

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
```

配置基于 AWS CRT 的 HTTP 客户端

基于 AWS CRT 的 HTTP 客户端包括同步 AwsCrtHttpClient 和异步 AwsCrtAsyncHttpClient。基于 AWS 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
- 缩短了延迟时间
- 连接运行状况管理
- DNS 负载均衡

SDK 中基于 AWS CRT 的组件

本主题中介绍的基于 AWS CRT 的 HTTP 客户端，与基于 AWS CRT 的 S3 客户端是 SDK 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。

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，具有多种优点。

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 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，基于 AWS CRT 的 S3 客户端是专为支持 S3 分段上传 API 功能而构建的。

访问基于 AWS CRT 的 HTTP 客户端

在使用基于 AWS CRT 的 HTTP 客户端之前，请先将版本最低为 2.22.0 的 `aws-crt-client` 构件添加到项目的依赖项中。

使用以下选项之一来设置 Maven `pom.xml` 文件。

> **Note**
>
> 如果您需要缩小运行时依赖项的大小（例如，如果您的应用程序在 AWS Lambda 函数中运行），则可以选择使用平台特定的 jar 选项。

Uber-jar option

默认情况下，`aws-crt-client` 使用一个包含 Linux、Windows 和 macOS 等多个平台二进制文件的 AWS CRT 构件的全能 JAR 包。

```
<project>
   <properties>
     <aws.sdk.java.version>2.29.10*</aws.sdk.java.version>
  </properties>
  <dependencyManagement>
   <dependencies>
      <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>bom</artifactId>
        <version>${aws.sdk.version}</version>
```

```
        <type>pom</type>
        <scope>import</scope>
      </dependency>
   </dependencies>
  </dependencyManagement>
  <dependencies>
   <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>aws-crt-client</artifactId>
   </dependency>
  </dependencies>
</project>
```

*将以红色显示的版本替换为您要使用的 Java SDK 版本。在 Maven Central 上查找最新版本。

Platform-specific jar option

要将 Java 运行时限制为特定于平台的 AWS CRT 库版本，请对全能 JAR 包选项进行以下更改。

- 向 SDK 的 `aws-crt-client` 构件添加一个 `exclusions` 元素。此排除操作可防止 SDK 以传递依赖方式使用 AWS CRT 的全能 JAR 包。
- 为您所需的特定 AWS CRT 平台版本添加依赖项元素。有关如何确定正确版本的信息，请参阅下面的确定 AWS CRT 构件版本的步骤。

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

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

[1]将以红色显示的版本替换为您要使用的 Java SDK 版本。在 Maven Central 上查找最新版本。

[2]替换全能 JAR 包选项将提供的 `software.amazon.awssdk.crt:aws-crt` 版本。请参阅下面确定 AWS CRT 构件版本的步骤。

[3]将 `classifier` 值替换为适用于您平台的值。有关可用值的列表，请参阅适用于 Java 的 AWS CRT GitHub 页面。

确定 AWS CRT 构件版本的步骤

使用以下步骤来确定与您正在使用的适用于 Java 的 SDK 版本兼容的 AWS CRT 构件版本。

1. 按照全能 JAR 包选项所示设置您的 `pom.xml` 文件。借助此设置，您可以查看默认情况下 SDK 会引入什么版本的 `software.amazon.awssdk.crt:aws-crt`。
2. 在项目的根目录下（与 `pom.xml` 文件位于同一目录中），运行以下 Maven 命令：

   ```
   mvn dependency:tree -Dincludes=software.amazon.awssdk.crt:aws-crt
   ```

   Maven 可能会执行其他操作，但最后您应该会看到 SDK 以传递依赖方式使用的 `software.amazon.awssdk.crt:aws-crt` 依赖项的控制台输出。以下代码段显示了基于 SDK 版本 `2.29.10` 的示例输出：

   ```
   [INFO] org.example:yourProject:jar:1.0-SNAPSHOT
   [INFO] \- software.amazon.awssdk:aws-crt-client:jar:2.29.10:compile
   [INFO]    \- software.amazon.awssdk.crt:aws-crt:jar:0.31.3:compile
   ```

3. 使用控制台为 `software.amazon.awssdk.crt:aws-crt` 构件显示的版本。在本例中，将 `0.31.3` 添加到您的 `pom.xml` 文件中。

使用和配置基于 AWS CRT 的 HTTP 客户端

您可以在生成服务客户端的同时配置一个基于 AWS 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。

无论哪种方法，您都可以使用生成器为基于 AWS CRT 的 HTTP 客户端实例配置属性。

最佳实践：将一个实例专用于一个服务客户端

如果您需要配置基于 AWS CRT 的 HTTP 客户端的实例，我们建议您与服务客户端一同构建，从而将实例专用。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基于 AWS CRT 的 HTTP 客户端实例却不关闭实例时可能发生的内存泄漏。

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

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和内存使用量，您可以配置一个基于 AWS 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**
>
> 共享基于 AWS CRT 的 HTTP 客户端实例时，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。

以下示例使用 `connectionTimeout` 和 `maxConcurrency` 值配置基于 AWS 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.
```

将基于 AWS CRT 的 HTTP 客户端设置为默认客户端

您可以设置 Maven 构建文件，让 SDK 使用基于 AWS 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，SDK 将基于 AWS 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>
   <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

要使用基于 AWS CRT 的 HTTP 客户端作为应用程序的默认 HTTP 客户端，可以将 Java 系统属性 software.amazon.awssdk.http.async.service.impl 的值设置为 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**
>
> 当您使用系统属性配置基于 AWS CRT 的 HTTP 客户端的使用时，需要在 poml.xml 文件中向 aws-crt-client 构件添加依赖项。

基于 AWS CRT 的 HTTP 客户端的高级配置

您可以使用基于 AWS 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 方法为基于 AWS 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，该客户端使用基于 AWS CRT 的 HTTP 客户端实例，配置了连接运行状况配置和连接最大空闲时间。

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

基于 AWS 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该支持。

在现阶段，如果您使用的是需要 HTTP/2 支持的服务客户端，例如 KinesisAsyncClient 或 TranscribeStreamingAsyncClient，可以考虑改用 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

您可以通过使用代码、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 服务客户端所用的基于 Apache 的 HTTP 客户端的示例代理配置。

```
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。

- Apache HTTP 客户端
- 基于 URLConnection 的 HTTP 客户端
- 基于 Netty 的 HTTP 客户端
- 基于 AWS CRT 的 HTTP 客户端

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` 生成器，SDK 也会查找外部设置来配置默认代理配置。

默认情况下，SDK 先搜索 JVM 系统属性。即使只找到一个属性，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 时此列表也有效。	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
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 URL	全部
HTTPS_PROXY[1]	采用 HTTPS 架构的有效 URL	Netty、CRT

环境变量	描述	HTTP 客户端支持
NO_PROXY[2]	应绕过代理直接访问的主机列表。该列表对 HTTP 和 HTTPS 均有效。	全部

查看要点和脚注

全部 - SDK 提供全部 HTTP 客户端（UrlConnectionHttpClient、ApacheHttpClient、NettyNioAsyncHttpClient、AwsCrtAsyncH

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

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

[1]查询的环境变量（HTTP_PROXY 或 HTTPS_PROXY）取决于客户端的 ProxyConfiguration 中的架构设置。默认架构是 HTTP。以下代码段说明如何将用于环境变量解析的架构更改为 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。主机名可能包括“*”通配符。

使用多种设置的组合

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- 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 `useEnvironmentVariablesValues` 的设置。默认情况下，两个设置均为 `true`。当设置为 `true` 时，SDK 会自动将来自系统属性或环境变量的值用于 `ProxyConfiguration` 生成器未提供的选项。

> **Important**
>
> 系统属性优先于环境变量。如果找到 HTTP 代理系统属性，SDK 将从系统属性中检索全部值，而不会从环境变量中检索任何值。如果要使环境变量优先于系统属性，请将 `useSystemPropertyValues` 设置为 `false`。

在本示例中，以下设置在运行时可用：

```
// 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`，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 解析的代理设置包括：

```
Host = ENV_VAR_HOST
Port = 3333
Password = EnvironmentPassword
UserName = EnvironmentUser
```

```
Non ProxyHost = environmentnonproxy.host, environmentnonproxy2.host:1234
```

配置基于 Apache 5.x 的 HTTP 客户端

访问 Apache5 HttpClient

要使用，Apache5HttpClient您必须添加对服务客户端的依赖关系apache5-client并在服务客户端Apache5HttpClient上进行显式配置。

```
<dependencyManagement>
   <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>bom</artifactId>
            <version>2.41.0*</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>

<dependencies>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>s3</artifactId>
    </dependency>

    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>apache5-client</artifactId>
    </dependency>
</dependencies>
```

*将以红色显示的版本替换为您要使用的 Java SDK 版本。在 Maven Central 上查找最新版本。

使用和配置 **Apache5HttpClient**

您可以在生成服务客户端的同时配置一个 Apache5HttpClient 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。

无论哪种方法，您都可以使用 Apache5. HttpClient Builder 为基于 Apache 5 的 HTTP 客户端配置属性。

最佳实践：将 Apache5 HttpClient 实例专用于服务客户端

如果您需要配置 Apache5HttpClient 实例，建议您生成专用 Apache5HttpClient 实例。您可以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 Apache5HttpClient 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，并使用 maxConnections 和 ConnectionTimeout 值配置Apache5HttpClient的嵌入式实例。HTTP 实例是使用 S3Client.Builder 的 httpClientBuilder 方法创建的。

导入

```
import software.amazon.awssdk.http.apache5.Apache5HttpClient;
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(Apache5HttpClient.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 **Apache5HttpClient** 实例

为了帮助降低应用程序的资源和内存使用量，您可以配置 Apache5HttpClient 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 Apache5HttpClient 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.apache5.Apache5HttpClient;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.s3.S3Client;
```

代码

```
SdkHttpClient apache5HttpClient = Apache5HttpClient.builder()
        .maxConnections(100).build();

// Singletons: Use the s3Client and dynamoDbClient for all requests.
S3Client s3Client =
    S3Client.builder()
            .httpClient(apache5HttpClient).build();

DynamoDbClient dynamoDbClient =
    DynamoDbClient.builder()
                  .httpClient(apache5HttpClient).build();

// Perform work with the s3Client and dynamoDbClient.

// Requests completed: Close all service clients.
s3Client.close();
dynamoDbClient.close();
apache5HttpClient.close();  // Explicitly close apache5HttpClient.
```

在AWS SDK for Java 2.x 中使用执行拦截器

AWS SDK for Java 2.x 中的执行拦截器可介入请求和响应生命周期，在 API 调用执行的各个阶段执行自定义逻辑。使用拦截器来实施通用功能，例如日志记录、指标收集、请求修改、调试和错误处理。

拦截器实施了 `ExecutionInterceptor` 接口，该接口为请求执行的不同阶段提供钩子。

拦截器生命周期

`ExecutionInterceptor` 接口提供了在请求执行期间的特定时刻调用的方法：

- beforeExecution - 在请求执行之前调用
- modifyRequest - 修改 SDK 请求对象
- beforeMarshalling - 在请求封送到 HTTP 之前调用
- afterMarshalling - 在请求封送到 HTTP 之后调用
- modifyHttpRequest - 修改 HTTP 请求
- beforeTransmission - 在 HTTP 请求发送之前调用
- afterTransmission - 在收到 HTTP 响应之后调用
- modifyHttpResponse - 修改 HTTP 响应
- beforeUnmarshalling - 在 HTTP 响应解组之前调用
- afterUnmarshalling - 在 HTTP 响应解组之后调用
- modifyResponse - 修改 SDK 响应对象
- afterExecution - 请求执行成功之后调用
- onExecutionFailure - 请求执行失败时调用

注册拦截器

使用 overrideConfiguration 方法构建服务客户端时注册拦截器。您可以注册多个拦截器，它们按照注册的顺序执行。

```
// Register a single interceptor.
SqsClient client = SqsClient.builder()
    .overrideConfiguration(c -> c.addExecutionInterceptor(new LoggingInterceptor()))
    .build();

// Register multiple interceptors.
S3Client s3Client = S3Client.builder()
    .overrideConfiguration(config -> config
        .addExecutionInterceptor(new TimingInterceptor())
        .addExecutionInterceptor(new LoggingInterceptor())
        .addExecutionInterceptor(new RequestModificationInterceptor()))
    .build();
```

拦截器示例

以下类演示了如何在不更改核心业务逻辑的情况下使用执行拦截器将诸如日志记录、性能监控和请求修改之类的通用功能添加到 S3 操作中。

此示例向您展示如何在 S3 客户端上注册多个拦截器，并在实际 AWS API 调用期间查看它们的具体运用。

导入

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.core.ApiName;
import software.amazon.awssdk.core.SdkRequest;
import software.amazon.awssdk.core.interceptor.Context;
import software.amazon.awssdk.core.interceptor.ExecutionAttributes;
import software.amazon.awssdk.core.interceptor.ExecutionInterceptor;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.SdkHttpResponse;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;

import java.time.Duration;
import java.time.Instant;
```

```
public class S3InterceptorsDemo {
    private static final Logger logger =
 LoggerFactory.getLogger(S3InterceptorsDemo.class);

    public static void main(String[] args) {
        logger.info("=== AWS SDK for Java v2 - S3 Interceptors Demo ===");

        // Create an S3 client with multiple interceptors.
        S3Client s3Client = S3Client.builder()
            .overrideConfiguration(config -> config
                .addExecutionInterceptor(new TimingInterceptor())
                .addExecutionInterceptor(new LoggingInterceptor())
                .addExecutionInterceptor(new RequestModificationInterceptor()))
            .build();

        try {
            logger.info("Starting S3 operations with interceptors...");

            // Operation 1: List buckets.
            logger.info("Operation 1: Listing S3 buckets");
            logger.info("----------------------------------------");
```

```
            ListBucketsResponse listBucketsResponse = s3Client.listBuckets();
            logger.info("Found {} buckets", listBucketsResponse.buckets().size());

            // Operation 2: Try to access a bucket that likely doesn't exist.
            logger.info("Operation 2: Checking non-existent bucket (demonstrating error
interceptor)");
            logger.info("----------------------------------------");
            try {
                s3Client.headBucket(HeadBucketRequest.builder()
                    .bucket("amzn-s3-demo-bucket-that-does-not-exist-1234")
                    .build());
            } catch (Exception e) {
                logger.info("Expected error occurred (interceptor should have logged
it)");
            }

        } catch (Exception e) {
            logger.error(" Error during S3 operations: {}", e.getMessage(), e);
        } finally {
            s3Client.close();
            logger.info("Demo completed - S3 client closed");
        }
    }

    // Logging interceptor.
    private static class LoggingInterceptor implements ExecutionInterceptor {
        private static final Logger logger =
LoggerFactory.getLogger(LoggingInterceptor.class);

        @Override
        public void beforeExecution(Context.BeforeExecution context,
ExecutionAttributes executionAttributes) {
            logger.info("[LOGGING] Starting request: {}",
context.request().getClass().getSimpleName());
        }

        @Override
        public void afterExecution(Context.AfterExecution context, ExecutionAttributes
executionAttributes) {
            logger.info("[LOGGING] Completed request: {}",
context.request().getClass().getSimpleName());
        }

        @Override
```

```
        public void onExecutionFailure(Context.FailedExecution context,
ExecutionAttributes executionAttributes) {
            logger.error("[LOGGING] Request failed: {}",
context.request().getClass().getSimpleName());
            if (context.exception() instanceof AwsServiceException) {
                AwsServiceException ase = (AwsServiceException) context.exception();
                if (ase.awsErrorDetails().errorCode() != null) {
                    SdkHttpResponse httpResponse =
ase.awsErrorDetails().sdkHttpResponse();
                    logger.error("   HTTP Status: {}", httpResponse.statusCode());
                    logger.error("   Error Code: {}",
ase.awsErrorDetails().errorCode());
                    logger.error("   Error Message: {}",
ase.awsErrorDetails().errorMessage());
                }
            }
        }
    }

    // Performance timing interceptor.
    private static class TimingInterceptor implements ExecutionInterceptor {
        private static final Logger logger =
LoggerFactory.getLogger(TimingInterceptor.class);
        private Instant startTime;

        @Override
        public void beforeExecution(Context.BeforeExecution context,
ExecutionAttributes executionAttributes) {
            startTime = Instant.now();
            logger.info("##  [TIMING] Request started at: {}", startTime);
        }

        @Override
        public void afterExecution(Context.AfterExecution context, ExecutionAttributes
executionAttributes) {
            if (startTime != null) {
                Duration duration = Duration.between(startTime, Instant.now());
                logger.info("##  [TIMING] Request completed in: {}ms",
duration.toMillis());
            }
        }

        @Override
```

```
        public void onExecutionFailure(Context.FailedExecution context,
ExecutionAttributes executionAttributes) {
            if (startTime != null) {
                Duration duration = Duration.between(startTime, Instant.now());
                logger.warn("##  [TIMING] Request failed after: {}ms",
duration.toMillis());
            }
        }
    }

    // Request modification interceptor
    private static class RequestModificationInterceptor implements ExecutionInterceptor
{
        private static final Logger logger =
LoggerFactory.getLogger(RequestModificationInterceptor.class);

        @Override
        public SdkRequest modifyRequest(Context.ModifyRequest context,
ExecutionAttributes executionAttributes) {
            SdkRequest originalRequest = context.request();
            logger.info("[MODIFY] Modifying request: {}",
originalRequest.getClass().getSimpleName());

            // For ListBucketsRequest, we can't modify much since it has no settable
properties
            // For HeadBucketRequest, we can demonstrate modifying the request
            if (originalRequest instanceof HeadBucketRequest) {
                HeadBucketRequest headRequest = (HeadBucketRequest) originalRequest;

                // Create a new request with an API name added.
                return HeadBucketRequest.builder()
                        .bucket(headRequest.bucket())
                        .overrideConfiguration(b -> b.addApiName(ApiName.builder()
                                .name("My-API")
                                .version("1.0")
                                .build()))
                        .build();
            }
            logger.info("Not a HeadBucketRequest, returning original request");
            return originalRequest;
        }

        @Override
```

```
        public SdkHttpRequest modifyHttpRequest(Context.ModifyHttpRequest context,
 ExecutionAttributes executionAttributes) {
            logger.info("[MODIFY] Adding custom HTTP headers");
            return context.httpRequest().toBuilder()
                .putHeader("X-Custom-Header", "S3InterceptorDemo")
                .putHeader("X-Request-ID", java.util.UUID.randomUUID().toString())
                .build();
        }
    }
}
```

Maven pom 文件

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

    <groupId>org.example</groupId>
    <artifactId>interceptors-examples</artifactId>
    <version>1.0.0</version>
    <packaging>jar</packaging>

    <name>interceptors-examples</name>
    <description>Demonstration of execution interceptors in AWS SDK for Java v2</
description>

    <properties>
        <maven.compiler.source>17</maven.compiler.source>
        <maven.compiler.target>17</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <aws.java.sdk.version>2.31.62</aws.java.sdk.version>
        <exec.mainClass>org.example.S3InterceptorsDemo</exec.mainClass>
    </properties>

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

```
                <type>pom</type>
                <scope>import</scope>
            </dependency>
            <dependency>
                <groupId>org.apache.logging.log4j</groupId>
                <artifactId>log4j-bom</artifactId>
                <version>2.23.1</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>s3</artifactId>
        </dependency>

        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-core</artifactId>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
            <version>2.0.13</version>
        </dependency>
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-slf4j2-impl</artifactId>
        </dependency>
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-1.2-api</artifactId>
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter</artifactId>
            <version>5.10.1</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
```

```
    <build>
        <plugins>
            <!-- Compiler Plugin -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.11.0</version>
                <configuration>
                    <source>17</source>
                    <target>17</target>
                </configuration>
            </plugin>

            <!-- Surefire Plugin for running tests -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>3.2.2</version>
            </plugin>

            <!-- Exec Plugin for running the main class -->
            <plugin>
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>exec-maven-plugin</artifactId>
                <version>3.1.0</version>
                <configuration>
                    <mainClass>${exec.mainClass}</mainClass>
                </configuration>
            </plugin>

            <!-- Shade Plugin to create executable JAR -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-shade-plugin</artifactId>
                <version>3.4.1</version>
                <executions>
                    <execution>
                        <phase>package</phase>
                        <goals>
                            <goal>shade</goal>
                        </goals>
                        <configuration>
                            <transformers>
```

```
                                    <transformer
 implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">
                                        <mainClass>${exec.mainClass}</mainClass>
                                    </transformer>
                                </transformers>
                                <createDependencyReducedPom>false</
createDependencyReducedPom>
                            </configuration>
                        </execution>
                    </executions>
                </plugin>
            </plugins>
        </build>
</project>
```

最佳实践

- 使拦截器保持轻量化 - 拦截器会为每个请求执行，因此请避免进行繁重的计算或阻止可能影响性能的操作。
- 优雅地处理异常 - 如果您的拦截器抛出异常，则会导致整个请求失败。对于可能失败的操作，请务必使用 try-catch 块。
- 顺序很重要 - 拦截器按照注册的顺序执行。注册拦截器时，请考虑拦截器之间的依赖项。
- 使用 ExecutionAttributes 来保存状态 - 如果您需要在不同拦截器方法之间传递数据，请使用 `ExecutionAttributes` 而不是实例变量来确保线程安全。
- 注意敏感数据 - 在记录请求和响应时，请注意不要记录敏感信息，例如凭证或个人数据。

上下文对象

每个拦截器方法都会接收一个上下文对象，该对象在不同执行阶段用于访问请求和响应信息：

- `Context.BeforeExecution` - 用于访问原始 SDK 请求
- `Context.ModifyRequest` - 修改 SDK 请求
- `Context.ModifyHttpRequest` - 修改 HTTP 请求
- `Context.AfterExecution` - 用于访问请求和响应
- `Context.FailedExecution` - 用于访问请求和发生的异常

使用 AWS SDK for Java 2.x

本章介绍了如何高效使用AWS SDK for Java 2.x。了解创建服务客户端、发出请求、处理响应和管理错误。本章介绍了同步和异步编程、分页结果、资源监视 Waiter 以及性能优化。

您还将找到客户端重复使用、故障排除指导、Lambda 启动优化、HTTP/2 支持和 DNS 配置的最佳实践。

目录

- 使用AWS SDK for Java 2.x 发出 AWS 服务请求
  - 使用服务客户端发出请求
    - 创建服务客户端
    - 默认客户端配置
    - 配置服务客户端
    - 关闭服务客户端
  - 提出请求
    - 使用请求来覆盖客户端配置
  - 处理回复
- 使用AWS SDK for Java 2.x 异步编程
  - 使用异步客户端 API
  - 使用异步方法处理流
  - 配置高级异步选项
- 使用AWS SDK for Java 2.x 的最佳实践
  - 通过配置 API 超时来防止请求挂起
  - 通过重复使用服务客户端来提高性能
  - 通过关闭未使用的服务客户端来防止资源泄漏
  - 通过关闭输入流来防止连接池耗尽
  - 针对您的应用程序工作负载优化 HTTP 性能
  - 使用适用于异步客户端的 OpenSSL 提高 SSL 性能
  - 使用 SDK 指标监控应用程序性能
- 在AWS SDK for Java 2.x 中处理错误
  - 为什么使用未选中的异常？

- AwsServiceException（和子类）
- SdkClientException
- 异常和重试行为
- 使用AWS SDK for Java 2.x 中的分页结果
  - 同步分页
    - 迭代页面
    - 迭代对象
      - 使用流
      - 使用 for-each 循环
    - 手动分页
  - 异步分页
    - 迭代表名称页面
      - 使用 Subscriber
      - 使用 Consumer
    - 迭代表名称
      - 使用 Subscriber
      - 使用 Consumer
    - 使用第三方库
- 在AWS SDK for Java 2.x 中使用 Waiter
  - 先决条件
  - 使用 Waiter
    - 同步编程
    - 异步编程
  - 配置 Waiter
    - 配置 Waiter
    - 覆盖特定请求的配置
  - 代码示例
- 问题排查常见问题
- 如何修复“java.net.SocketException: Connection reset”或“server failed to complete the response”错误？

- 如何修复“连接超时”？
- 如何修复“java.net.SocketTimeoutException: Read timed out”？
- 如何修复“Unable to execute HTTP request: Timeout waiting for connection from pool”错误？
- 我该如何修复 NoClassDefFoundError、NoSuchMethodError 或 NoSuchFieldError？
- 如何修复“SignatureDoesNotMatch”错误或“The request signature we calculated does not match the signature you provided”错误？
- 如何修复“java.lang.IllegalStateException: Connection pool shut down”错误？
- 如何修复“Unable to load credentials from any of the providers in the chain AwsCredentialsProviderChain”？
  - 常见原因和解决方案
    - 检查您的凭证配置
      - 对于 Amazon EC2 实例
      - 对于容器环境
      - 对于本地开发
      - 对于 Web 身份联合验证
    - 网络或代理连接问题
- 为 AWS Lambda 缩短 SDK 启动时间
  - 使用基于 AWS CRT 的 HTTP 客户端
  - 移除未使用的 HTTP 客户端依赖项
  - 配置服务客户端以进行快捷查找
  - 在 Lambda 函数处理程序之外初始化 SDK 客户端
  - 尽量减少依赖关系注入
  - 使用针对 AWS Lambda 的 Maven 原型
  - 考虑适用于 Java 的 Lambda SnapStart
  - 影响启动时间的 2.x 版更改
  - 其他资源
- 在AWS SDK for Java 2.x 中实施 ContentStreamProvider
  - 使用 mark() 和 reset()
  - 如果 mark() 和 reset() 不可用，则使用缓冲
  - 创建新流
- 为 DNS 名称查找设置 JVM TTL

- 如何设置 JVM TTL
- 在 AWS SDK for Java 中使用 HTTP/2

使用AWS SDK for Java 2.x 发出 AWS 服务请求

使用服务客户端发出请求

完成设置 SDK 中的步骤并了解如何配置服务客户端后，您就可以向诸如 Amazon S3、Amazon DynamoDB、AWS Identity and Access Management、Amazon EC2 等 AWS 服务发出请求了。

创建服务客户端

要向 AWS 服务 发出请求，必须先使用静态工厂方法 `builder()` 为该服务实例化服务客户端。`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 服务都需要在服务客户端生成器中指定 `Region`；但是，最佳实践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默认 AWS 区域提供程序链。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

有关如何配置服务客户端的详细信息，请参阅通过外部方式配置客户端和代码中的客户端配置。

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 AWS Security Token Service 的 `StsClient` 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();
    }
}
```

提出请求

使用服务客户端向对应的 AWS 服务 提出请求。

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 fluent API。借助 fluent API，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));
```

使用请求来覆盖客户端配置

尽管服务客户端是不可变的，但您可以在请求级别覆盖它的许多设置。在构建请求时，您可以使用 AwsRequestOverrideConfiguration 实例来提供覆盖设置。可以用来覆盖客户端设置的一些方法包括：

- `apiCallAttemptTimeout`
- `apiCallTimeout`
- `credentialProvider`
- `compressionConfiguration`
- `putHeader`

查看使用请求覆盖客户端设置的示例时，假设您有以下使用默认设置的 S3 客户端。

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

您想下载一个大文件，并希望确保在下载完成之前请求不会超时。为此，请仅增大单个 `GetObject` 请求的超时值，如以下代码所示。

Standard API

```
AwsRequestOverrideConfiguration overrideConfiguration =
 AwsRequestOverrideConfiguration.builder()
    .apiCallTimeout(Duration.ofSeconds(100L))
    .apiCallAttemptTimeout(Duration.ofSeconds(25L))
    .build();

GetObjectRequest request = GetObjectRequest.builder()
    .bucket("amzn-s3-demo-bucket")
    .key("demo-key")
    .overrideConfiguration(overrideConfiguration)
    .build();

s3Client.getObject(request, myPath);
```

Fluent API

```
s3Client.getObject(b -> b
        .bucket("amzn-s3-demo-bucket")
        .key("demo-key")
        .overrideConfiguration(c -> c
            .apiCallTimeout(Duration.ofSeconds(100L))
            .apiCallAttemptTimeout(Duration.ofSeconds(25L))),
    myPath);
```

处理回复

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求处理响应对象中的信息。

例如，以下代码段输出上一个请求中随 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 Simple Email Service 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());
}
```

使用AWS SDK for Java 2.x 异步编程

AWS SDK for Java 2.x 提供了异步客户端，这些客户端支持非阻塞 I/O，可以在几个线程中实现高并发。然而，并不能保证完全实现非阻塞 I/O。异步客户端在某些情况下仍可能执行阻塞调用，这些情况包括凭证检索、使用AWS签名版本 4（SigV4）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，所以需要通过某种方法在响应准备就绪时接收响应。AWS SDK for Java 2.x 中的异步客户端方法将返回 CompletableFuture 对象，这些对象可让您在响应准备就绪时访问响应。

使用异步客户端 API

异步客户端方法的签名与其对应同步方法的签名相同，但是异步方法返回 CompletableFuture 对象，该对象在未来会包含异步操作的结果。如果在 SDK 的异步方法执行时发生错误，则错误会作为 `CompletionException` 引发。

您可以用来获取结果的一种方法是，将一个 `whenComplete()` 方法链接到 SDK 方法调用所返回的 `CompletableFuture`。`whenComplete()` 方法接收结果或 Throwable 类型的对象 `CompletionException`，具体取决于完成异步调用的方式。您向 `whenComplete()` 提供一个操作，用于在结果返回给调用代码之前，对结果进行处理或检查。

如果要返回的不是 SDK 方法原本返回的对象，而是其他内容，请改用 `handle()` 方法。`handle()` 方法采用的参数与 `whenComplete()` 相同，但您可以处理结果并返回对象。

要等待异步链完成并检索完成结果，可以调用 join() 方法。如果未在链中处理 Throwable 对象，则 join() 方法会抛出封装了原始异常的 CompletionException。您可以使用 CompletionException#getCause() 访问原始异常。您也可以调用 CompletableFuture#get() 方法来获取完成结果。但 get() 方法会抛出受检异常。

以下示例显示了两种变体，说明如何使用 DynamoDB 异步客户端的 listTables() 方法。传递给 whenComplete() 的操作仅记录成功的响应，而 handle() 版本则提取表名列表并返回列表。在这两种情况下，如果异步链中生成错误，则会重新抛出错误，以便客户端代码有机会对其进行处理。

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

whenComplete() variation

```
public class DynamoDbAsyncListTables {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        DynamoDbAsyncClient dynamoDbAsyncClient =
 DynamoDbAsyncClient.builder().region(region).build();
        try {
            ListTablesResponse listTablesResponse =
 listTablesWhenComplete(dynamoDbAsyncClient).join();  // The join() method may throw
 a CompletionException.
            if (listTablesResponse.hasTableNames()){
                System.out.println("Table exist in this region: " + region.id());
            }
        } catch (RuntimeException e) {
            // Handle as needed. Here we simply print out the class names.
            System.out.println(e.getClass()); // Prints 'class
 java.util.concurrent.CompletionException'.
            System.out.println(e.getCause().getClass()); // Prints 'class
 software.amazon.awssdk.services.dynamodb.model.DynamoDbException'.
```

```
        }
    }

    public static CompletableFuture<ListTablesResponse>
listTablesWhenComplete(DynamoDbAsyncClient client) {
        return client.listTables(ListTablesRequest.builder().build())
            .whenComplete((listTablesResponse, throwable) -> {
                if (listTablesResponse != null) {  // Consume the response.
                    System.out.println("The SDK's listTables method completed
successfully.");
                } else {
                    RuntimeException cause = (RuntimeException)
throwable.getCause(); // If an error was thrown during the SDK's listTables method
it is wrapped in a CompletionException.

 // The SDK throws only RuntimeExceptions, so this is a safe cast.
                    System.out.println(cause.getMessage());  // Log error here, but
rethrow so the calling code can handle as needed.
                    throw cause;
                }
            });
    }
```

handle() variation

```
public class DynamoDbAsyncListTables {

    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        DynamoDbAsyncClient dynamoDbAsyncClient =
DynamoDbAsyncClient.builder().region(region).build();
        try {
            List<String> tableNames =
listTablesHandle(dynamoDbAsyncClient).join(); // The join() method may throw a
CompletionException.
            tableNames.forEach(System.out::println);
        } catch (RuntimeException e) {
            // Handle as needed. Here we simply print out the class names.
            System.out.println(e.getClass()); // Prints 'class
java.util.concurrent.CompletionException'.
            System.out.println(e.getCause().getClass()); // Prints 'class
software.amazon.awssdk.services.dynamodb.model.DynamoDbException'.
        }
```

```
    }

    public static CompletableFuture<List<String>>
 listTablesHandle(DynamoDbAsyncClient client) {
        return client.listTables(ListTablesRequest.builder().build())
            .handle((listTablesResponse, throwable) -> {
                if (listTablesResponse != null) {
                    return listTablesResponse.tableNames(); // Return the list of
 table names.
                } else {
                    RuntimeException cause = (RuntimeException)
 throwable.getCause(); // If an error was thrown during the SDK's listTables method
 it is wrapped in a CompletionException.

  // The SDK throws only RuntimeExceptions, so this is a safe cast.
                    System.out.println(cause.getMessage());  // Log error here, but
 rethrow so the calling code can handle as needed.
                    throw cause;
                }
            });
    }
}
```

使用异步方法处理流

对于包含流式内容的异步方法，必须提供 AsyncRequestBody，以便以递增方式提供内容，或者提供 AsyncResponseTransformer 以接收和处理响应。

以下示例使用 PutObject 操作的异步形式，以异步方式将文件上传到 Amazon S3。

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 操作的异步形式，从 Amazon S3 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();
    }
}
```

配置高级异步选项

AWS SDK for Java 2.x 使用 Netty 处理 I/O 线程，这是一种异步的事件驱动网络应用程序框架。AWS SDK for Java 2.x 在 Netty 之后创建 `ExecutorService`，以完成从 HTTP 客户端请求返回到 Netty 客户端的 future。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。

在构建异步客户端时，可以指定特定的 JDK 实现 `ExecutorService`。以下代码段创建了一个具有固定线程数的 `ExecutorService`。

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该执行程序。

```
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();
```

使用AWS SDK for Java 2.x 的最佳实践

通过配置 API 超时来防止请求挂起

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。

通过重复使用服务客户端来提高性能

每个服务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。所有服务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。

通过关闭未使用的服务客户端来防止资源泄漏

如果不再需要某个服务客户端，请关闭该服务客户端以释放线程等资源。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。

通过关闭输入流来防止连接池耗尽

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。

针对您的应用程序工作负载优化 HTTP 性能

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。

使用适用于异步客户端的 OpenSSL 提高 SSL 性能

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();
```

使用 SDK 指标监控应用程序性能

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。

在AWS SDK for Java 2.x 中处理错误

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，这样做就会出现问题。

AwsServiceException（和子类）

AwsServiceException 是您在使用AWS SDK for Java 时会遇到的最常见异常。AwsServiceException 是更通用的 SdkServiceException 的子类。AwsServiceException 表示来自 AWS 服务的错误响应。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。这可能是因为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
- 来自 AwsErrorDetails 类的服务详细错误消息
- 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。有关 AwsServiceException 的 Java SDK API 参考中显示了大量 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，请查看 AwsErrorDetails 对象上的 errorCode。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 指示在尝试将请求发送到 AWS 或者在尝试解析来自 AWS 的响应时，Java 客户端代码内出现问题。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。

使用AWS SDK for Java 2.x 中的分页结果

当响应对象太大而无法在单个响应中返回时，很多 AWS 操作都会返回分页的结果。在AWS SDK for Java 1.0 中，响应包含了您用于检索下一页结果的令牌。与此不同的是，AWS SDK for Java 2.x 中采用的是自动分页方法，该方法可进行多个服务调用以自动为您获取下一页结果。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**
>
> 这些代码段假设您了解使用 SDK 的基础知识，并且已为环境配置了单点登录访问权限。

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

第一个示例演示如何使用 listRes 分页工具对象以及 ListObjectsV2Iterable 实例，借助 stream 方法迭代所有响应页面。代码流式通过响应页面，将响应流转换为 S3Object 内容的流，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 GitHub 上的完整示例。

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 GitHub 上的完整示例。

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 GitHub 上的完整示例。

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 GitHub 上的完整示例。

异步分页

以下示例演示列出 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` 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。要了解有关反应式流模型的更多信息，请参阅反应式流 GitHub 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 GitHub 上的完整示例。

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 GitHub 上的完整示例。

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 GitHub 上的完整示例。

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 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。本示例演示如何使用 RxJava，但实现反应式流接口的任何库都可以使用。有关该库的更多信息，请参阅 GitHub 上的 RxJava wiki 页面。

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

POM 条目

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

请参阅 GitHub 上的完整示例。

在AWS SDK for Java 2.x 中使用 Waiter

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

waiter 是一个抽象，用于轮询 AWS 资源，例如 DynamoDB 表或 Amazon S3 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）。编写逻辑来持续轮询 AWS 资源可能很繁琐且容易出错，而使用 waiter 来轮询资源，则可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，然后才能在AWS SDK for Java 项目中使用 waiter。

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>2.27.21</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

以下代码段演示了如何等待 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

以下代码段演示了如何等待 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.waitUntilTableNotExists(b -> b.tableName("myTable"),
               o -> o.maxAttempts(10));

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

代码示例

有关将 waiter 与 DynamoDB 结合使用的完整示例，请参阅 AWS 代码示例存储库中的 CreateTable.java。

有关将 waiter 与 Amazon S3 结合使用的完整示例，请参阅 AWS 代码示例存储库中的 S3BucketOps.java。

问题排查常见问题

在应用程序中使用AWS SDK for Java 2.x 时，您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找出根本原因并解决错误。

如何修复“`java.net.SocketException`: Connection reset”或“server failed to complete the response”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。这种情况在流式操作中很常见：由于一段时间内没有任何数据通过网络进行读写，中间方检测到连接已经失效并将其关闭。为防止出现这种情况，请确保您的应用程序会主动下载或上传数据。
- 您关闭了 HTTP 或 SDK 客户端。确保不要在资源仍在使用时将其关闭。
- 代理配置错误。尝试绕过您配置的任何代理，以查看这样是否可以解决问题。如果这样可以解决问题，则表明代理出于某种原因正在关闭您的连接。请研究您的特定代理以确定代理关闭连接的原因。

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

如果您看到 AWS 端点正在发送 TCP RST（重置），请联系受影响服务的支持人员，看看他们能否确定重置的原因。请准备好提供请求 ID 和发生问题时的时间戳。AWS 支持团队可能还需要网络传输日志，这些日志能精确显示您的应用程序在何时收发了哪些字节数据。

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时间内与服务器建立新连接。以下项目列出了导致此问题的常见原因。

- 配置的连接超时太低。默认情况下，AWS SDK for Java 2.x 中的连接超时为 2 秒。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调用，则建议的连接超时为 1 秒；如果您发出跨区域请求，则建议的连接超时为 3 秒。
- 代理配置错误。尝试绕过您配置的任何代理，以查看这样是否可以解决问题。如果这样可以解决问题，则表明连接超时的根源在于代理。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（例如，在您控制的任何代理之后）为受影响的连接运行 TCP 转储，以便调查是否存在任何网络问题。

如何修复“`java.net.SocketTimeoutException`: Read timed out”？

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、AWS 服务或任何中间方（例如，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，就必须知道为什么没有返回数据。

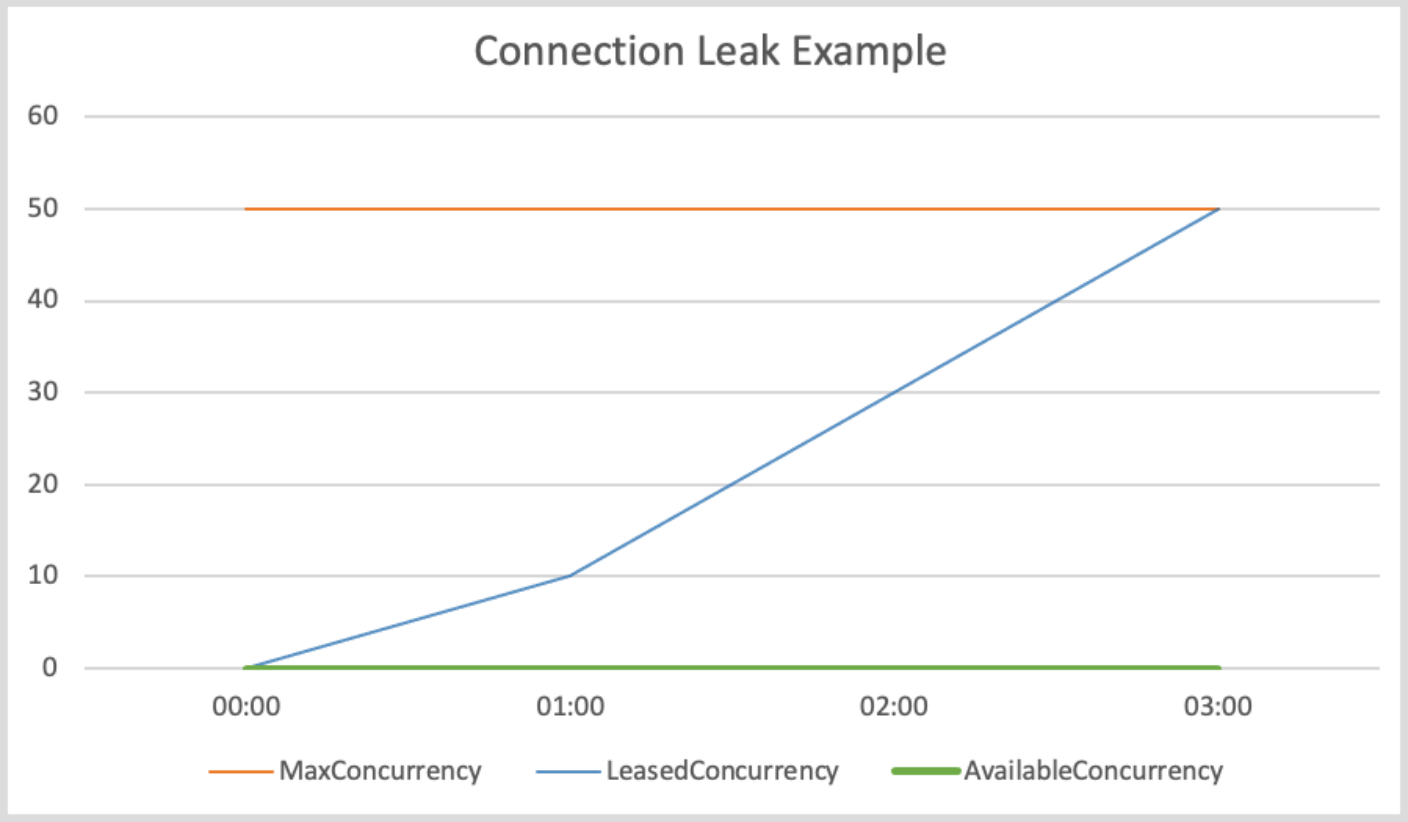
尝试在网络的客户端边缘（例如，在您控制的任何代理之后）为受影响的连接运行 TCP 转储。

如果您看到 AWS 端点正在发送 TCP RST（重置），请联系受影响服务的支持人员。请准备好提供请求 ID 和发生问题时的时间戳。AWS 支持团队可能还需要网络传输日志，这些日志能精确显示您的应用程序在何时收发了哪些字节数据。

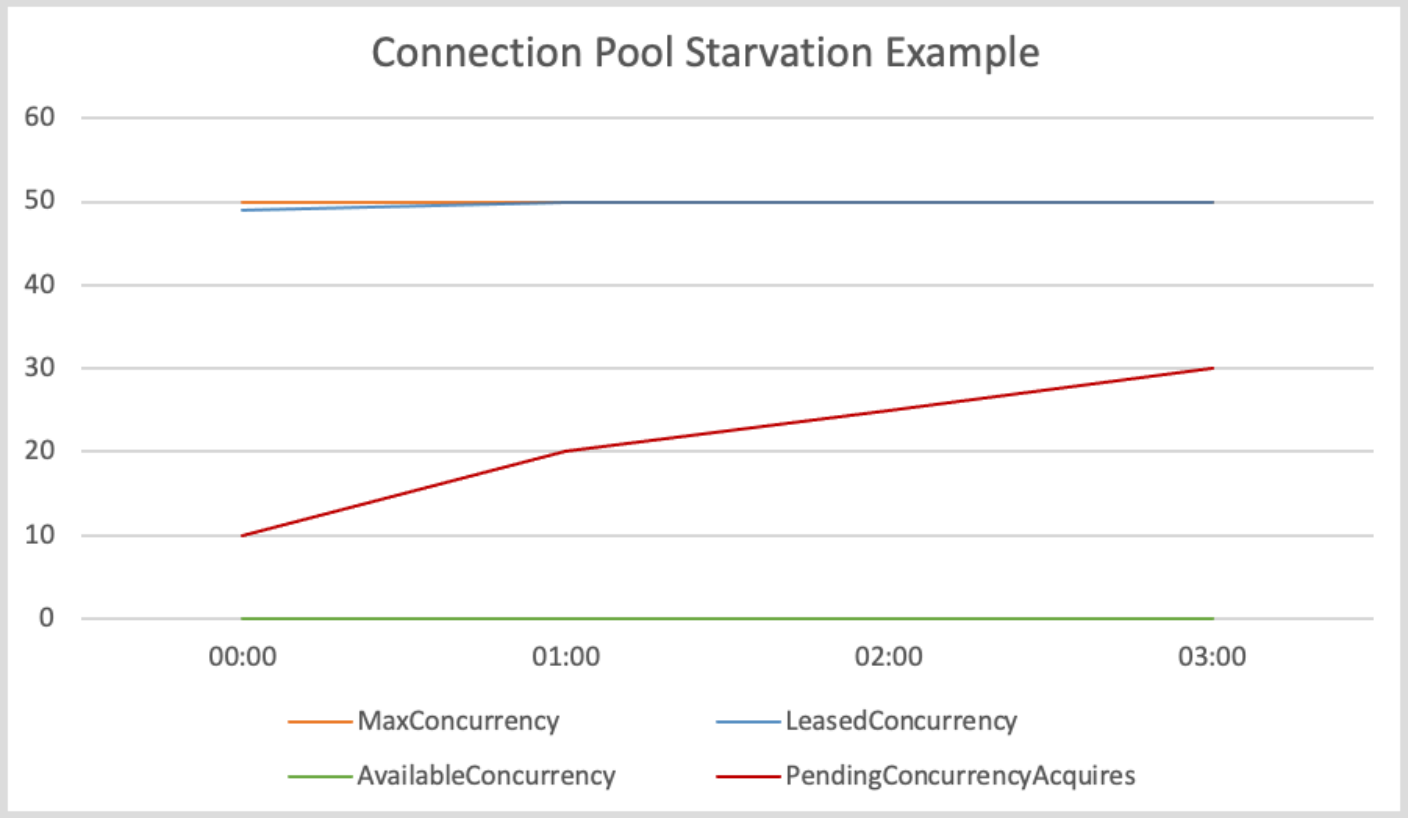
如何修复“Unable to execute HTTP request: Timeout waiting for connection from pool”错误？

此错误表示请求无法在指定的最长时间内从池中获得连接。要解决此问题，我们建议您启用 SDK 客户端指标，以便将指标发布到 Amazon CloudWatch。HTTP 指标有助于缩小根本原因范围。以下项目列出了导致此错误的常见原因。

- 连接泄露。您可以通过检查 `LeasedConcurrency`、`AvailableConcurrency` 和 `MaxConcurrency` 指标来调查此问题。如果 `LeasedConcurrency` 持续增长，直至达到 `MaxConcurrency` 但从未下降，则可能存在连接泄露。发生泄露的常见原因是流式操作（例如 S3 `getObject` 方法）未关闭。我们建议您的应用程序尽快读取输入流中的所有数据，之后关闭输入流。下图显示了连接泄露的 SDK 指标看起来可能是什么样子。



- 连接池匮乏。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入队列，直到连接可用为止。下图显示了连接池匮乏的 SDK 指标看起来可能是什么样子。



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。
- 增加获取超时。
- 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（除非网络接口已充分利用）。但是，您最终可能会达到操作系统对进程所使用的文件描述符数量施加的限制。如果您已经充分使用网络接口或无法进一步增加连接数，请尝试增大获取超时时间。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。

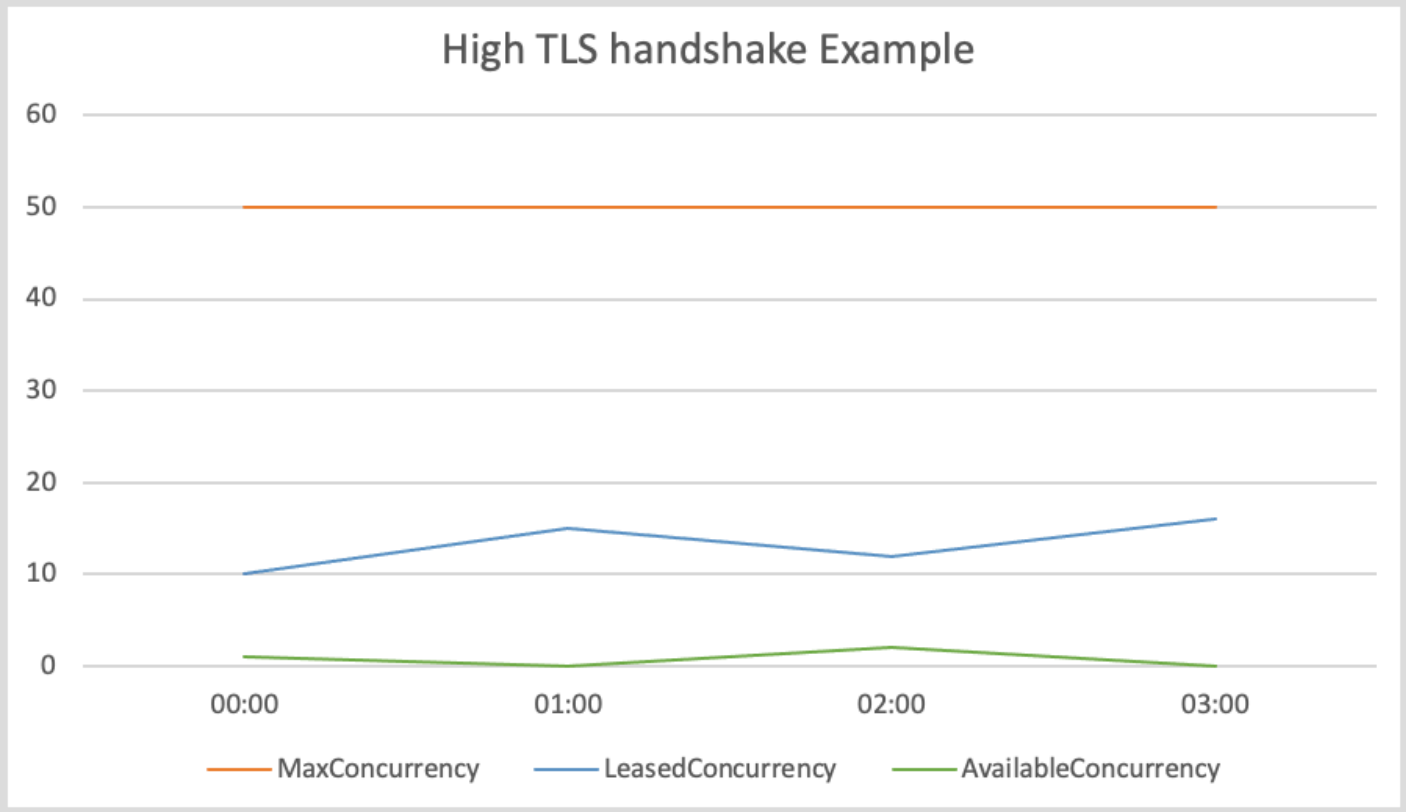
如果您无法通过前两种机制解决问题，请尝试通过以下选项来降低请求速率。

- 将您的请求流量平滑化，避免突发的大流量冲击导致客户端过载。
- 更高效地调用 AWS 服务。
- 增加发送请求的主机数量。

- I/O 线程太忙。这仅适用于使用基于 `NettyNioAsyncHttpClient` 的异步 SDK 客户端的情况。如果 `AvailableConcurrency` 指标不低（表示池中存在连接），但 `ConcurrencyAcquireDuration` 很高，则可能是因为 I/O 线程无法处理请求。确保您未将

Runnable:run 作为未来完成执行器来传递，并且未在响应未来完成链中执行耗时的任务，因为这会阻塞 I/O 线程。如果情况并非如此，请考虑使用 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 小于 MaxConcurrency，则可能是因为 TLS 握手延迟很高。下图显示了 TLS 握手延迟高的 SDK 指标可能是什么样子。



对于 Java SDK 提供的未基于 CRT 的 HTTP 客户端，请尝试启用 TLS 日志来解决 TLS 问题。对于基于 AWS 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，则应联系受影响服务的支持人员。

我该如何修复 NoClassDefFoundError、NoSuchMethodError 或 NoSuchFieldError？

NoClassDefFoundError 表示在运行时无法加载某个类。出现此错误的两个最常见原因是：

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

同样，NoSuchMethodError 和 NoSuchFieldError 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. 检查您的依赖项，确保使用的所有 SDK JAR 包版本一致。找不到类、方法或字段的最常见原因是，您升级到新的客户端版本，但继续使用旧的“共享”SDK 依赖项版本。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。

   下面的示例演示混合使用 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
   ```

   dynamodb 的版本是 2.20.00，aws-core 的版本是 2.13.19。aws-core 构件版本也应为 2.20.00。
2. 在日志的早期部分检查相关语句，以查看某个类是否因静态初始化失败而无法加载。当类第一次加载失败时，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，因此后续日志语句只会报告未找到该类。
3. 检查您的部署过程，确保所需的 JAR 文件已随应用程序一同部署。您可能确实使用了正确的版本进行构建，但为应用程序创建类路径的过程却排除了一个必需的依赖项。

如何修复“SignatureDoesNotMatch”错误或“The request signature we calculated does not match the signature you provided”错误？

SignatureDoesNotMatch 错误表示AWS SDK for Java 生成的签名和AWS 服务 生成的签名不匹配。以下项目说明了潜在原因。

- 代理或中间方修改请求。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 在各自生成待签名字符串时，对请求进行编码的方式不一致。

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，待签名字符串用 AWS4 String to sign: ... 标记。

如果您无法启用调试（例如，因为错误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，尝试在生产环境之外（例如，在启用了调试日志记录的集成测试中）复现该错误。

收集规范请求和待签名字符串后，将其与 AWS 签名版本 4 规范进行比较，以确定 SDK 生成待签名字符串的方式是否存在问题。如果出现问题，可以创建 GitHub 错误报告并发送给 AWS SDK for Java。

如果没有出现任何问题，您可以将 SDK 的待签名字符串与某些 AWS 服务（例如 Amazon S3）在失败响应中返回的待签名字符串进行比较。如果未提供此信息，则应联系受影响服务的支持人员，以查看他们生成了哪些规范请求和待签名字符串，以便进行比较。这些比较有助于识别是否有中间方可能修改了请求，或发现服务与客户端之间在编码处理上的差异。

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待签名字符串的

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

如何修复“**java.lang.IllegalStateException**: Connection pool shut down”错误？

此错误表示底层 Apache HTTP 连接池已关闭。以下项目说明了潜在原因。

- SDK 客户端过早关闭。只有在关闭了关联客户端时，SDK 才会关闭连接池。确保不要在资源仍在使用时将其关闭。
- 出现 **java.lang.Error** 错误。OutOfMemoryError 等错误会导致 Apache HTTP 连接池关闭。检查您的日志，找出错误的堆栈跟踪信息。还要检查您的代码中是否存在捕获了 Throwable 或 Error 但却“吞掉”输出，从而导致错误无法暴露出来的地方。如果您的代码未报告错误，请重写代码，以便可以记录信息。记录的信息有助于确定错误的根本原因。
- 在 **DefaultCredentialsProvider#create()** 返回的凭证提供程序关闭后您仍然尝试使用它。DefaultCredentialsProvider#create 返回一个单例实例，因此，如果它已关闭且您的代码调用了 resolveCredentials 方法，则在缓存的凭证（或令牌）过期后会引发异常。

  检查您的代码中是否存在关闭了 DefaultCredentialsProvider 的地方，如下面的示例中所示。

  - 通过调用 DefaultCredentialsProvider#close(). 来关闭单例实例。

```
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 块中调用 DefaultCredentialsProvider#create()。

```
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();
```

如果您的代码已关闭单例实例，并且您需要使用 `DefaultCredentialsProvider` 解析凭证，则通过调用 `DefaultCredentialsProvider.builder().build()` 来创建新的非单例实例。

如何修复“Unable to load credentials from any of the providers in the chain AwsCredentialsProviderChain”？

此错误表示AWS SDK for Java 2.x 无法通过默认凭证提供程序链中的任何凭证提供程序找到有效的 AWS 凭证。SDK 会自动按特定顺序搜索凭证，当链中的所有提供程序都无法提供有效凭证时，就会发生此错误。

完整的错误消息通常如下所示（为了提高可读性，添加了行尾结束符号和缩进）：

```
Unable to load credentials from any of the providers in the chain
 AwsCredentialsProviderChain(
    credentialsProviders=[
        SystemPropertyCredentialsProvider(),
        EnvironmentVariableCredentialsProvider(),
        WebIdentityTokenCredentialsProvider(),
        ProfileCredentialsProvider(profileName=default,
 profileFile=ProfileFile(sections=[])),
        ContainerCredentialsProvider(),
        InstanceProfileCredentialsProvider()
    ]) : [
        SystemPropertyCredentialsProvider(): Unable to load credentials from system
 settings.
        Access key must be specified either via environment variable
 (AWS_ACCESS_KEY_ID)
```

```
        or system property (aws.accessKeyId).,

        EnvironmentVariableCredentialsProvider(): Unable to load credentials from
system settings.
        Access key must be specified either via environment variable
(AWS_ACCESS_KEY_ID)
        or system property (aws.accessKeyId).,

        WebIdentityTokenCredentialsProvider(): To use web identity tokens, the 'sts'
service module
        must be on the class path.,

        ProfileCredentialsProvider(profileName=default,
profileFile=ProfileFile(sections=[])):
        Profile file contained no credentials for profile 'default':
ProfileFile(sections=[]),

        ContainerCredentialsProvider(): Cannot fetch credentials from container -
neither
        AWS_CONTAINER_CREDENTIALS_FULL_URI or AWS_CONTAINER_CREDENTIALS_RELATIVE_URI
environment
        variables are set.,

        InstanceProfileCredentialsProvider(): Failed to load credentials from IMDS.]
```

常见原因和解决方案

检查您的凭证配置

当您使用默认凭证提供程序（在没有显式配置凭证的情况下调用 `ServiceClient.create()`）时，SDK 会按特定顺序搜索凭证。查看默认凭证提供程序链的工作原理，以了解 SDK 会检查哪些凭证来源以及按什么顺序检查。

确保在您的环境中正确设置了您打算使用的凭证配置方法：

对于 Amazon EC2 实例

- 检查 IAM 角色：验证 IAM 角色是否已附加到您的实例。
- IMDS 间歇性故障：如果您遇到间歇性故障（通常持续几百毫秒），这通常表示实例元数据服务（IMDS）出现暂时性网络问题。

  解决方案：

- 启用调试日志记录以分析发生故障的时间和频率
- 考虑在应用程序中实施重试逻辑以应对与凭证相关的故障
- 检查您的实例和 IMDS 端点之间的网络连接问题

对于容器环境

确认已配置任务角色（Amazon ECS）或服务账户（Amazon EKS），并已设置所需的环境变量。

对于本地开发

确认已设置好环境变量、凭证文件或 IAM Identity Center 配置。

对于 Web 身份联合验证

- 验证配置：确认存在 Web 身份令牌文件且已配置所需的环境变量。
- 缺少 STS 模块依赖项：如果您看到错误 `To use web identity tokens, the 'sts' service module must be on the class path`，则需要将 STS 模块添加为依赖项。在使用 Amazon EKS 容器组身份或其他 Web 身份令牌身份验证时，这种情况很常见。

  解决方案：将 STS 模块添加到您的项目依赖项中：

  ```
  <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>sts</artifactId>
  </dependency>
  ```

  对于某些服务，您可能还需要 `aws-query-protocol` 依赖项：

  ```
  <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>aws-query-protocol</artifactId>
  </dependency>
  ```

网络或代理连接问题

如果您在凭证提供程序链中看到 `Connection refused` 错误，这通常表示 SDK 尝试访问 AWS 端点时存在网络连接问题。

解决方案：

- 如果正在使用代理服务器，请验证代理配置
- 确认您的网络是否允许向 AWS 端点发起出站 HTTPS 连接
- 启用调试日志记录以查看详细的连接尝试
- 使用 `curl` 等工具测试连接，以便验证 AWS 端点的网络访问

为 AWS Lambda 缩短 SDK 启动时间

AWS SDK for Java 2.x 的目标之一是降低 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。

使用基于 AWS CRT 的 HTTP 客户端

为了与 AWS Lambda 配合使用，我们建议在同步场景中使用 `AwsCrtHttpClient`，在异步场景中使用 `AwsCrtAsyncHttpClient`。

本指南中的 the section called “配置基于 AWS CRT 的 HTTP 客户端” 主题描述了使用 HTTP 客户端的好处、如何添加依赖项以及如何按服务客户端配置其使用。

移除未使用的 HTTP 客户端依赖项

除了明确使用基于 AWS CRT 的客户端之外，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。（使用基于 AWS CRT 的客户端时不需要这些客户端。）此示例从 S3 客户端依赖项中排除 HTTP 客户端构件，并添加 `aws-crt-client` 构件以允许访问基于 AWS CRT 的 HTTP 客户端。

```
<project>
    <properties>
        <aws.java.sdk.version>2.27.21</aws.java.sdk.version>
    <properties>
    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>software.amazon.awssdk</groupId>
                <artifactId>bom</artifactId>
```

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

> **Note**
>
> 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。这可简化 SDK 的默认区域查找过程，该过程会在多个位置查找 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 凭证查找过程的时间。

> **Note**
>
> 利用此凭证提供程序，代码能够在 Lambda 函数中使用，但可能不适用于 Amazon EC2 或其他系统。
> 如果您打算在某个时候使用适用于 Java 的 Lambda SnapStart，则应依靠默认的凭证提供程序链来查找凭证。如果您指定 `EnvironmentVariableCredentialsProvider`，则初始凭证查找会起作用，但是当激活 SnapStart 时，Java 运行时会设置容器凭证环境变量。激活后，`EnvironmentVariableCredentialsProvider` 使用的环境变量（访问密钥环境变量）对 Java SDK 不可用。

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。

使用针对 AWS Lambda 的 Maven 原型

AWS Java SDK 团队开发了一个 Maven 原型模板，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。

考虑适用于 Java 的 Lambda SnapStart

如果您的运行时系统要求兼容，则 AWS 提供了适用于 Java 的 Lambda SnapStart。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。当您发布函数的新版本时，Lambda SnapStart 会对其进行初始化，并拍摄内存和磁盘状态的不可变的加密快照。然后，SnapStart 会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 开发人员指南》中有一节介绍开发非特定于 Java 的 Lambda 函数的最佳实践。

有关通过 AWS Lambda 使用 Java 构建云原生应用程序的示例，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。

在AWS SDK for Java 2.x 中实施 **ContentStreamProvider**

`ContentStreamProvider` 是AWS SDK for Java 2.x 中使用的抽象，用于允许多次读取输入数据。本主题将介绍如何为应用程序正确实施 `ContentStreamProvider`。

适用于 Java 的 SDK 2.x 每次需要读取整个流时使用 `ContentStreamProvider#newStream()` 方法。要让此方法适用于整个流，返回的流必须始终位于内容的起始位置，并且必须包含相同的数据。

在接下来的章节中，我们将提供三种正确实现该行为的方法。

使用 **mark()** 和 **reset()**

在下面的示例中，我们在读取开始之前在构造函数中使用 `mark(int)`，以确保我们可以将流重置回开头。对于 `newStream()` 的每次调用，我们都会重置流：

```
public class MyContentStreamProvider implements ContentStreamProvider {
    private InputStream contentStream;

    public MyContentStreamProvider(InputStream contentStream) {
        this.contentStream = contentStream;
        this.contentStream.mark(MAX_LEN);
    }

    @Override
    public InputStream newStream() {
        contentStream.reset();
        return contentStream;
    }
}
```

如果 mark() 和 reset() 不可用，则使用缓冲

如果您的流不直接支持 mark() 和 reset()，您仍然可以使用前面显示的解决方案，方法是先将流封装在 BufferedInputStream 中：

```
public class MyContentStreamProvider implements ContentStreamProvider {
    private BufferedReader contentStream;

    public MyContentStreamProvider(InputStream contentStream) {
        this.contentStream = new BufferedInputStream(contentStream);
        this.contentStream.mark(MAX_LEN);
    }

    @Override
    public InputStream newStream() {
        contentStream.reset();
        return contentStream;
    }
}
```

创建新流

一种更简单的方法是：每次调用时都直接获取一个全新的数据流，并关闭前一个数据流：

```
public class MyContentStreamProvider implements ContentStreamProvider {
    private InputStream contentStream;

    @Override
    public InputStream newStream() {
        if (contentStream != null) {
            contentStream.close();
        }
        contentStream = openStream();
        return contentStream;
    }
}
```

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，它会在指定时间段内 (称为生存时间 (TTL)) 缓存 IP 地址。

由于 AWS 资源使用偶尔变更的 DNS 名称条目，因此建议您为 JVM 配置 5 秒的 TTL 值。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 AWS 资源的 IP 地址在应用程序仍在运行时发生更改，则在您手动重新启动 JVM 并刷新缓存的 IP 信息之前，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 networkaddress.cache.ttl 安全属性值，在 Java 8 的 `$JAVA_HOME/jre/lib/security/java.security` 文件中或者在 Java 11 或更高版本的 `$JAVA_HOME/conf/security/java.security` 文件中设置 `networkaddress.cache.ttl` 属性。

以下是 `java.security` 文件中的代码段，该代码段显示 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 `$JAVA_HOME` 环境变量所代表的 JVM 上运行的所有应用程序都会使用此设置。

在 AWS SDK for Java 中使用 HTTP/2

HTTP/2 是 HTTP 协议的一个主要修订。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）可让客户端和 AWS 之间的多个请求和响应消息同时通过单一连接（而不是通过多个连接）进行传输，这样可以提高性能。

升级到最新开发工具包的开发人员将自动使用 HTTP/2（如果他们使用的服务支持）。新编程接口无缝地利用 HTTP/2 功能并提供新的方法来构建应用程序。

AWS SDK for Java 2.x 提供了新的实施 HTTP/2 协议的 API 进行事件流式传输。有关如何使用这些新 API 的示例，请参阅使用 Kinesis。

从AWS SDK for Java 2.x 调用 AWS 服务

本部分提供有关如何使用精选 AWS 服务的简短教程和指导。如需完整的示例集合，请参阅代码示例部分。

主题

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

与... 一起工作 CloudWatch

本节提供了使用 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) 资源和您运行 AWS 的应用程序。您可以使用 CloudWatch 来收集和跟踪指标，这些指标是您可以衡量资源和应用程序的变量。CloudWatch 警报会根据您定义的规则发送通知或自动更改您正在监控的资源。

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。您可以在那里下载单个源文件，也可以将存储库复制到本地以获得所有示例，然后构建并运行这些示例。

主题

- 从中获取指标 CloudWatch
- 将自定义指标数据发布到 CloudWatch
- 处理 CloudWatch 警报
- 使用 Amazon CloudWatch Events

从中获取指标 CloudWatch

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**
>
> AWS 服务发布的指标和维度列表可在 Amazon CloudWatch 用户指南的Amazon CloudWatch 指标和维度参考中找到。

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 服务在以 “AWS” 开头的命名空间中发布自己的指标。您也可以使用自己的命名空间发布自定义指标数据（只要不是以 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产品使用。 Amazon Web Services

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。
- AWS 《 Amazon CloudWatch 用户指南》中的@@ 命名空间。
- PutMetricData在 Amazon CloudWatch API 参考中。

处理 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's `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 desCWAlarms( CloudWatchClient cw) {

        try {

            boolean done = false;
            String newToken = null;

            while(!done) {
                DescribeAlarmsResponse response;

                if (newToken == null) {
                    DescribeAlarmsRequest request =
DescribeAlarmsRequest.builder().build();
                    response = cw.describeAlarms(request);
                } else {
                    DescribeAlarmsRequest request = DescribeAlarmsRequest.builder()
                        .nextToken(newToken)
                        .build();
                    response = cw.describeAlarms(request);
                }

                for(MetricAlarm alarm : response.metricAlarms()) {
                    System.out.printf("\n Retrieved alarm %s", alarm.alarmName());
                }

                if(response.nextToken() == null) {
                    done = true;
```

```
                } else {
                    newToken = response.nextToken();
                }
            }

        } catch (CloudWatchException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.printf("Done");
    }
```

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 Events

CloudWatch Events 提供几乎实时的系统事件流，这些事件描述 AWS 资源中对 Amazon EC2 实例、Lambda 函数、Kinesis 流、Amazon ECS 任务、Step Functions 状态机、Amazon SNS 主题、Amazon SQS 队列或内置目标的更改。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 Events 的演进版本。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 Events 客户端，也可以迁移到适用于 Java 的 SDK EventBridge 客户端以使用 CloudWatch Events 功能。CloudWatch Events 用户指南文档和 API 参考现已通过 EventBridge 文档网站提供。

添加事件

要添加自定义 CloudWatch 事件，请使用包含一个或多个 `PutEventsRequestEntry` 对象（提供每个事件的详细信息）的 `PutEventsRequest` 对象调用 `CloudWatchEventsClient`'s `putEvents` 方法。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 GitHub 上的完整示例。

添加规则

要创建或更新规则，请使用包含规则名称和可选参数的 PutRuleRequest 调用 CloudWatchEventsClient's putRule 方法，可选参数包括事件模式、与规则相关联的 IAM 角色以及描述规则运行频率的计划表达式。

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

请参阅 GitHub 上的完整示例。

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。

要向规则添加目标，请使用 PutTargetsRequest（包含要更新的规则和要添加到规则的目标列表）来调用 CloudWatchEventsClient's 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 GitHub 上的完整示例。

更多信息

- 《Amazon EventBridge 用户指南》中的适用 PutEvents 添加事件
- 《Amazon EventBridge 用户指南》中的规则的计划表达式
- 《Amazon EventBridge 用户指南》中的 CloudWatch Events 的事件类型
- 《Amazon EventBridge 用户指南》中的事件模式
- 《Amazon EventBridge API 参考》中的 PutEvents
- 《Amazon EventBridge API 参考》中的 PutTargets
- 《Amazon EventBridge API 参考》中的 PutRule

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 AWS 中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 Web 服务 API 以及用于与数据交互的 Web 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 具有相应的 DocDbClient 接口，用于处理资源。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	使用数据库
SDK service client: DynamoDbClient	SDK service client: DynamoDbClient
Example: 使用 DynamoDB 的 React/Spring REST 应用程序	Examples: CreateTable、ListTables、DeleteTable
Examples: 几个 DynamoDB 示例	

处理数据	使用数据库
SDK service client: DynamoDbEnhancedClient Example: 使用 DynamoDB 的 React/Spring REST 应用程序 Examples: 几个 DynamoDB 示例 (names starting with 'Enhanced")	

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 示例

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 示例	示例：几个 RedshiftClient 示例
示例：使用 RedshiftDataClient 的 React/Spring REST 应用程序	

Amazon 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 RdsDataClient 的 React/Spring REST 应用程序	示例：几个 RdsClient 示例

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）	

使用 DynamoDB

DynamoDB 是一项完全托管式 NoSQL 数据库服务，提供快速且可预测的性能，能够实现无缝扩展。本节向您展示如何使用AWS SDK for Java 2.x 来处理 DynamoDB。

选择您的 DynamoDB 客户端

SDK 提供了两种处理 DynamoDB 的主要方法：

低级别客户端（`DynamoDbClient`）

> 通过完全控制请求和响应，提供对 DynamoDB 操作的直接访问。当您需要精细控制或使用动态架构时，请使用此客户端。

增强型客户端（`DynamoDbEnhancedClient`）

通过 Java 对象和 DynamoDB 项目之间的自动映射，提供面向对象的编程。还提供了面向文档的功能，用于处理不遵循固定架构且类似 JSON 的数据。在处理定义明确的数据模型或文档类型数据时，请使用此客户端。

配置 DynamoDB 客户端

在使用 DynamoDB 之前，请配置您的客户端以获得更好的性能和可靠性。

了解 DynamoDB 重试行为

DynamoDB 客户端使用的默认最大重试次数为 8，该值高于其他 AWS 服务客户端。这种较高的重试次数有助于应对 DynamoDB 的分布式特性和临时容量限制。有关重试策略的更多信息，请参阅the section called “重试”。

使用基于账户的端点来优化性能

DynamoDB 提供基于 AWS 账户的端点，通过使用您的 AWS 账户 ID 来简化请求路由，从而提升性能。

要使用此功能，您需要使用AWS SDK for Java 2.x 的版本 2.28.4 或更高版本。您可以在 Maven Central 存储库中找到最新版本。支持的 SDK 版本会自动使用新的端点。

要选择退出基于账户的路由，请选择以下选项之一：

- 配置 DynamoDB 服务客户端，将 `AccountIdEndpointMode` 设置为 `DISABLED`。
- 设置环境变量。
- 设置 JVM 系统属性。
- 更新共享的 AWS 配置文件设置。

以下示例演示了如何通过配置 DynamoDB 服务客户端来禁用基于账户的路由：

```
DynamoDbClient.builder()
                .accountIdEndpointMode(AccountIdEndpointMode.DISABLED)
                .build();
```

有关其他配置选项的更多信息，请参阅《AWS SDK 和工具参考指南》中的基于账户的端点。

本主题涵盖的内容

以下各节将向您展示如何使用 DynamoDB：

- the section called “处理 DynamoDB 中的表” - 创建、描述、更新和删除表
- the section called “处理中的项目 DynamoDB” - 添加、检索和更新单个项目
- the section called “ 将对象映射到 DynamoDB 项目” - 在增强型客户端中使用对象映射和面向文档的数据

有关额外的 DynamoDB 代码示例，请参阅 AWS 代码示例库中的 DynamoDB 代码示例。

处理 DynamoDB 中的表

表是 DynamoDB 数据库中所有项目的容器。您必须先创建表，然后才能在 DynamoDB 中添加或删除数据。

对于每个表，您必须定义：

- 表名称，它对于您的账户和所在区域是唯一的。
- 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）。

  每个键值均有一个由 `ScalarAttributeType` 类枚举的关联数据类型。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 定价基于您为表设置的预置吞吐量值，因此您应只为表保留可能需要的容量。

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's createTable 方法来创建新的 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**
>
> 如果使用您所选名称的表已存在，则将引发 DynamoDbException。

创建具有简单主键的表

此代码创建一个表，该表只有一个属性，这个属性是表的简单主键。该示例为 CreateTableRequest 使用 AttributeDefinition 和 KeySchemaElement 对象。

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 GitHub 上的完整示例。

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 GitHub 上的完整示例。

列出表

您可以通过调用 DynamoDbClient's `listTables` 方法列出特定区域中的表。

> **Note**
>
> 如果您的账户和区域没有使用该名称的表，则将引发 `ResourceNotFoundException` 异常。

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);
                }
            } 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，每次调用将返回最多 100 个表 – 对返回的 `ListTablesResponse` 对象使用 `lastEvaluatedTableName` 可获得评估的上一个表。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 GitHub 上的完整示例。

描述表（获取相关信息）

使用 `DynamoDbClient's describeTable` 方法获取有关表的信息。

> **Note**
>
> 如果您的账户和区域没有使用该名称的表，则将引发 ResourceNotFoundException 异常。

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 describeDymamoDBTable(DynamoDbClient ddb,String tableName ) {

        DescribeTableRequest request = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

        try {
            TableDescription tableInfo =
                    ddb.describeTable(request).table();

            if (tableInfo != null) {
                System.out.format("Table name  : %s\n",
                        tableInfo.tableName());
                System.out.format("Table ARN   : %s\n",
                        tableInfo.tableArn());
                System.out.format("Status      : %s\n",
                        tableInfo.tableStatus());
                System.out.format("Item count  : %d\n",
                        tableInfo.itemCount().longValue());
                System.out.format("Size (bytes): %d\n",
                        tableInfo.tableSizeBytes().longValue());

                ProvisionedThroughputDescription throughputInfo =
```

```
                tableInfo.provisionedThroughput();
            System.out.println("Throughput");
            System.out.format("  Read Capacity : %d\n",
                    throughputInfo.readCapacityUnits().longValue());
            System.out.format("  Write Capacity: %d\n",
                    throughputInfo.writeCapacityUnits().longValue());

            List<AttributeDefinition> attributes =
                    tableInfo.attributeDefinitions();
            System.out.println("Attributes");

            for (AttributeDefinition a : attributes) {
                System.out.format("  %s (%s)\n",
                        a.attributeName(), a.attributeType());
            }
        }
    } catch (DynamoDbException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    System.out.println("\nDone!");
}
```

请参阅 GitHub 上的完整示例。

修改（更新）表

您可以通过调用 DynamoDbClient’s 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没有使用该名称的表，则将引发 ResourceNotFoundException 异常。

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 GitHub 上的完整示例。

删除表

要删除表，请调用 DynamoDbClient’s deleteTable 方法并提供表名称。

> **Note**
>
> 如果您的账户和区域没有使用该名称的表，则将引发 ResourceNotFoundException 异常。

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 GitHub 上的完整示例。

更多信息

- 《Amazon DynamoDB 开发人员指南》中的表处理准则
- 《Amazon DynamoDB 开发人员指南》中的处理 DynamoDB 中的表

处理中的项目 DynamoDB

在中 DynamoDB，项目是属性的集合，每个属性都有一个名称和一个值。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
- 使用《 Amazon DynamoDB 开发者指南》 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。您可以在代码中定义表与其相应数据类之间的关系。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
- 使用扩展自定义 DynamoDB 增强型客户端操作
- 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。

从数据类生成 **TableSchema**

`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。

您可以对 getter 或 setter 应用属性级注释，但不能同时应用两者。

> **Note**
>
> 对于封装在 JavaBean 中的值，通常使用术语 property。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 类展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 进行自检，生成数据类属性与 DynamoDB 属性之间的映射。

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

DynamoDbEnhancedClient 类或其异步对应类 DynamoDbEnhancedAsyncClient 是使用 DynamoDB 增强型客户端 API 的入口点。

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，您的代码具有可以使用 Customer 实例的 DynamoDB 表内存表示形式。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.enhanced.dynamodb.model.CreateTableEnhancedRequest;
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 Integer、String、BigDecimal 和 Instant。AttributeConverter 接口的已知实现类中显示了相关列表。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 JavaBean 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。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。

- 如果无法访问该属性的 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 方法时，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 null 值属性。

SDK 对 updateItem 请求的默认行为是从 DynamoDB 的项目中移除您在 updateItem 方法中提交的对象中设置为 null 的属性。如果您打算更新某些属性值并使其他属性值保持不变，则有两种选择。

- 在更改值之前（使用 getItem）检索项目。通过使用这种方法，SDK 会将所有更新的值和旧值提交给 DynamoDB。
- 在生成更新项目的请求时，使用 IgnoreNullsMode.SCALAR_ONLY 或 IgnoreNullsMode.MAPS_ONLY。两种模式都忽略对象中表示 DynamoDB 中标量属性的 null 值属性。本指南中的 the section called “更新包含复杂类型的项目” 主题包含有关 IgnoreNullsMode 值以及如何处理复杂类型的更多信息。

以下示例演示 updateItem() 方法的 ignoreNullsMode()。

```
    public static void updateItemNullsExample() {
        Customer customer = new Customer();
        customer.setCustName("CustomerName");
        customer.setEmail("email");
        customer.setId("1");
        customer.setRegistrationDate(Instant.now());

        logger.info("Original customer: {}", customer);

        // Put item with values for all attributes.
        try {
            customerAsyncDynamoDbTable.putItem(customer).join();
        } catch (RuntimeException rte) {
            logger.error("A exception occurred during putItem: {}",
rte.getCause().getMessage(), rte);
            return;
        }

        // Create a Customer instance with the same 'id' and 'email' values, but a
different 'name' value.
        // Do not set the 'registrationDate' attribute.
        Customer customerForUpdate = new Customer();
        customerForUpdate.setCustName("NewName");
        customerForUpdate.setEmail("email");
```

```
        customerForUpdate.setId("1");

        // Update item without setting the 'registrationDate' property and set
 IgnoreNullsMode to SCALAR_ONLY.
        try {
            Customer updatedWithNullsIgnored = customerAsyncDynamoDbTable.updateItem(b
 -> b
                            .item(customerForUpdate)
                            .ignoreNullsMode(IgnoreNullsMode.SCALAR_ONLY))
                    .join();
            logger.info("Customer updated with nulls ignored: {}",
 updatedWithNullsIgnored.toString());
        } catch (RuntimeException rte) {
            logger.error("An exception occurred during updateItem: {}",
 rte.getCause().getMessage(), rte);
            return;
        }

        // Update item without setting the registrationDate attribute and not setting
 ignoreNulls to true.
        try {
            Customer updatedWithNullsUsed =
 customerAsyncDynamoDbTable.updateItem(customerForUpdate)
                    .join();
            logger.info("Customer updated with nulls used: {}",
 updatedWithNullsUsed.toString());
        } catch (RuntimeException rte) {
            logger.error("An exception occurred during updateItem: {}",
 rte.getCause().getMessage(), rte);
        }
    }


// Logged lines.
Original customer: Customer [id=1, name=CustomerName, email=email,
 regDate=2024-10-11T14:12:30.222858Z]
Customer updated with nulls ignored: Customer [id=1, name=NewName, email=email,
 regDate=2024-10-11T14:12:30.222858Z]
Customer updated with nulls used: Customer [id=1, name=NewName, email=email,
 regDate=null]
```

DynamoDB 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，如以下代码段所示。

```
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，如以下代码段示例所示。

```
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.waitUntilTableExists(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 createdDate;
}
```

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 `DynamoDbEnhancedAsyncClient` 返回的响应则返回一个 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 DynamoDbTable 的 scan 方法来探索 PageIterable 接口。

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。

在注释行 2 之后显示的筛选表达式将返回的 ProductCatalog 项目限制为价格在 8.00 到 80.00 之间（含）的项目。

此示例还使用注释行 1 后面显示的 attributesToProject 方法排除 isbn 值。

在注释行 3 之后，scan 方法返回 PageIterable 对象 pagedResults。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 之后的版本对每个页面进行流式处理，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），因此提供了多种方法来处理结果。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 类中的 CountDownLatch 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

您可以使用 DynamoDB 增强型客户端来查询您的表并检索符合特定条件的多个项目。query() 方法使用在数据类中定义的 @DynamoDbPartitionKey 和可选 @DynamoDbSortKey 注释，根据主键值来查找项目。

query() 方法需要分区键值，并且可以选择接受排序键条件以进一步优化结果。与 scan API 一样，对于同步调用，查询会返回 PageIterable，对于异步调用，则返回 PagePublisher。

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，该主键由作为分区键的 **movie** 属性和作为排序键的 **actor** 属性组成。

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：keyEqual（在注释行 1 之后）和 sortGreaterThanOrEqualTo（在注释行 1a 之后）。

按分区键查询项目

keyEqual 实例匹配分区键值为 **movie01** 的项目。

此示例还在注释行 2 之后定义了一个筛选表达式，用于筛选掉任何没有 **actingschoolname** 值的项目。

QueryEnhancedRequest 组合了查询的键条件和筛选表达式。

```
    public static void query(DynamoDbTable movieActorTable) {

        // 1. Define a QueryConditional instance to return items matching a partition
 value.
        QueryConditional keyEqual = QueryConditional.keyEqualTo(b ->
 b.partitionValue("movie01"));
        // 1a. Define a QueryConditional that adds a sort key criteria to the partition
 value criteria.
        QueryConditional sortGreaterThanOrEqualTo =
 QueryConditional.sortGreaterThanOrEqualTo(b ->
 b.partitionValue("movie01").sortValue("actor2"));
```

```
        // 2. Define a filter expression that filters out items whose attribute value
 is null.
        final Expression filterOutNoActingschoolname =
 Expression.builder().expression("attribute_exists(actingschoolname)").build();

        // 3. Build the query request.
        QueryEnhancedRequest tableQuery = QueryEnhancedRequest.builder()
                .queryConditional(keyEqual)
                .filterExpression(filterOutNoActingschoolname)
                .build();
        // 4. Perform the query using the "keyEqual" conditional and filter expression.
        PageIterable<MovieActor> pagedResults = movieActorTable.query(tableQuery);
        logger.info("page count: {}", pagedResults.stream().count()); // Log  number of
 pages.

        pagedResults.items().stream()
                .sorted()
                .forEach(
                        item -> logger.info(item.toString()) // Log the sorted list of
 items.
                );
```

Example – 使用 keyEqual 查询条件的输出

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'}
```

按分区键和排序键查询项目

sortGreaterThanOrEqualTo QueryConditional 通过为大于或等于 actor2 的值添加排序键条件来细化分区键匹配（movie01）。

以 sort 开头的 QueryConditional 方法需要有分区键值才能匹配，并通过基于排序键值的比较来进一步细化查询。方法名称中的 Sort 并不意味着结果已排序，而是表示将使用排序键值进行比较。

在以下代码段中，我们更改了之前在注释行 3 之后显示的查询请求。此代码段将“keyEqual”查询条件替换为在注释行 1a 之后定义的“sortGreaterThanOrEqualTo”查询条件。以下代码还删除了筛选表达式。

```
        QueryEnhancedRequest tableQuery = QueryEnhancedRequest.builder()
                .queryConditional(sortGreaterThanOrEqualTo).build();
```

Example – 使用 sortGreaterThanOrEqualTo 查询条件的输出

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。因为我们移除了筛选条件，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和排序键值来指定要读取的项目。如果数据类有复合键，则必须同时提供分区键值和排序键值。

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 value and sort key value for the item.
                .addGetItem(b -> b.key(k ->
k.partitionValue("1").sortValue("cust1@orgname.org")))
                // 1b. Alternatively, supply a data class instances to provide the
primary key values.
                .addGetItem(customer2)
                .build();

        // 2. Build a batch to read from the MovieActor table.
        ReadBatch moveActorBatch = ReadBatch.builder(MovieActor.class)
                .mappedTableResource(movieActorTable)
                // 2a. Call consistentRead(Boolean.TRUE) for each item for the same
table.
                .addGetItem(b -> b.key(k ->
k.partitionValue("movie01").sortValue("actor1")).consistentRead(Boolean.TRUE))
                .addGetItem(b -> b.key(k ->
k.partitionValue("movie01").sortValue("actor4")).consistentRead(Boolean.TRUE))
                .build();
```

```
        // 3. Add ReadBatch objects to the request.
        BatchGetResultPageIterable resultPages = enhancedClient.batchGetItem(b ->
 b.readBatches(customerBatch, moveActorBatch));

        // 4. Retrieve the successfully requested items from each table.
        resultPages.resultsForTable(customerTable).forEach(item ->
 logger.info(item.toString()));
        resultPages.resultsForTable(movieActorTable).forEach(item ->
 logger.info(item.toString()));

        // 5. Retrieve the keys of the items requested but not processed by the
 service.
        resultPages.forEach((BatchGetResultPage pageResult) -> {
            pageResult.unprocessedKeysForTable(customerTable).forEach(key ->
 logger.info("Unprocessed item key: " + key.toString()));
            pageResult.unprocessedKeysForTable(movieActorTable).forEach(key ->
 logger.info("Unprocessed item key: " + key.toString()));
        });
    }
```

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目。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。名为 SubjectLastPostedDateIndex 的 GSI 使用 Subject 属性作为分区键，使用 LastPostedDateTime 属性作为排序键。名为 ForumLastPostedDateIndex 的 LSI 使用 ForumName 作为其分区键，使用 LastPostedDateTime 作为其排序键。

请注意，Subject 属性起着双重作用。它是主键的排序键，也是名为 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，如以下示例所示。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，您将创建一个在调用 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 class IndexScanExamples {
    private static Logger logger = LoggerFactory.getLogger(IndexScanExamples.class);

    public static List<MessageThread> queryUsingSecondaryIndices(String lastPostedDate,

 DynamoDbTable<MessageThread> threadTable) {
        // 1. Log the parameter value.
        logger.info("lastPostedDate value: {}", lastPostedDate);

        // 2. Create a QueryConditional whose sort key value must be greater than or
 equal to the parameter value.
        QueryConditional queryConditional =
 QueryConditional.sortGreaterThanOrEqualTo(qc ->
                qc.partitionValue("Forum02").sortValue(lastPostedDate));

        // 3. Specify the index name to query.
        final DynamoDbIndex<MessageThread> forumLastPostedDateIndex =
 threadTable.index("ForumLastPostedDateIndex");
```

```
        // 4. Perform the query using the QueryConditional object.
        final SdkIterable<Page<MessageThread>> pagedResult =
 forumLastPostedDateIndex.query(q -> q
                .queryConditional(queryConditional)
                // 5. Request three attribute in the results.
                .attributesToProject("forumName", "subject", "lastPostedDateTime"));

        List<MessageThread> collectedItems = new ArrayList<>();
        // 6. Iterate through pages response and sort the items.
        pagedResult.stream().forEach(page -> page.items().stream()
                .sorted(Comparator.comparing(MessageThread::getLastPostedDateTime))
                .forEach(mt -> {
                    // 7. Log the returned items and add the collection to return to
 the caller.
                    logger.info(mt.toString());
                    collectedItems.add(mt);
                }));
        return collectedItems;
    }
```

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，代码未投影消息属性。

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。它可以在数据对象和 AttributeValues 映射之间进行映射。`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。生成器方法避免使用注释，也不需要 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 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 中更新操作的示例包括 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 createdDate;

    @DynamoDbPartitionKey
    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getCreatedDate() { return createdDate; }
    public void setCreatedDate(String createdDate) { this.createdDate =
 createdDate; }
}
```

Lombok

Lombok 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 createdDate;
}
```

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 createdDate;

    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getCreatedDate() { return createdDate; }
    public void setCreatedDate(String createdDate) { this.createdDate =
 createdDate; }
```

Lombok

```
@Data
@ToString(callSuper = true)
public class Customer extends GenericRecord{
    private String name;
}
```

```
@Data
public abstract class GenericRecord {
    private String id;
    private String createdDate;
}
```

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 createdDate;

    @DynamoDbPartitionKey
    public String getId() { return this.id; }
    public void setId(String id) { this.id = id; }

    public String getCreatedDate() { return this.createdDate; }
```

```
    public void setCreatedDate(String createdDate) { this.createdDate =
 createdDate; }
}
```

Lombok

```
@Data
@DynamoDbBean
public class Customer {
    private String name;
    @Getter(onMethod_=@DynamoDbFlatten)
    private GenericRecord record;
}

@Data
@DynamoDbBean
public class GenericRecord {
    @Getter(onMethod_=@DynamoDbPartitionKey)
    private String id;
    private String createdDate;
}
```

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 createdDate;

    public String getId() { return this.id; }
    public void setId(String id) { this.id = id; }

    public String getCreatedDate() { return this.createdDate; }
    public void setCreatedDate(String createdDate) { this.createdDate =
 createdDate; }
}
```

Lombok

```
@Data
public class Customer {
    private String name;
    private GenericRecord record;
}

@Data
public class GenericRecord {
    private String id;
    private String createdDate;
}
```

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。

处理属性值为 bean、map、list 和 set 的对象

一个 bean 定义（例如下面所示的 `Person` 类）可能会定义一些属性，而这些属性所引用的类型本身还包含更多的属性。例如，在 `Person` 类中，`mainAddress` 是一个属性，该属性引用定义了额外值属性的 `Address` bean。`addresses` 引用一个 Java map，其元素引用 `Address` bean。这些复杂类型可以看作是包含简单属性的容器，在 DynamoDB 环境中，您使用这些容器是为了获取其所包含的实际数据值。

DynamoDB 将嵌套元素（例如 map、list 或 bean）的值属性称为嵌套属性。Amazon DynamoDB 开发人员指南将 Java 中的 map、list 或 bean 对象保存后的形式称为文档类型。在 Java 中，用于其数据值的简单属性在 DynamoDB 中称为标量类型。set 包含多个相同类型的标量元素，所以称为集类型。

重要的是要知道，DynamoDB 增强型客户端 API 在保存时会将 bean 属性转换为 DynamoDB map 文档类型。

`Person` 类

```
@DynamoDbBean
public class Person {
    private Integer id;
    private String firstName;
    private String lastName;
    private Integer age;
    private Address mainAddress;
    private Map<String, Address> addresses;
    private List<PhoneNumber> phoneNumbers;
    private Set<String> hobbies;

    @DynamoDbPartitionKey
    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getFirstName() {
        return firstName;
    }
```

```
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public Integer getAge() {
        return age;
    }

    public void setAge(Integer age) {
        this.age = age;
    }

    public Address getMainAddress() {
        return mainAddress;
    }

    public void setMainAddress(Address mainAddress) {
        this.mainAddress = mainAddress;
    }

    public Map<String, Address> getAddresses() {
        return addresses;
    }

    public void setAddresses(Map<String, Address> addresses) {
        this.addresses = addresses;
    }

    public List<PhoneNumber> getPhoneNumbers() {
        return phoneNumbers;
    }

    public void setPhoneNumbers(List<PhoneNumber> phoneNumbers) {
        this.phoneNumbers = phoneNumbers;
    }
```

```
    public Set<String> getHobbies() {
        return hobbies;
    }

    public void setHobbies(Set<String> hobbies) {
        this.hobbies = hobbies;
    }

    @Override
    public String toString() {
        return "Person{" +
               "addresses=" + addresses +
               ", id=" + id +
               ", firstName='" + firstName + '\'' +
               ", lastName='" + lastName + '\'' +
               ", age=" + age +
               ", mainAddress=" + mainAddress +
               ", phoneNumbers=" + phoneNumbers +
               ", hobbies=" + hobbies +
               '}';
    }
}
```

`Address` 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 + '\'' +
                '}';
    }
}
```

保存复杂类型

使用带注释的数据类

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);
```

在生成器中使用抽象架构

另一种方法是为每个嵌套 bean 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.documentOf(Address.class, TABLE_SCHEMA_ADDRESS),
a -> a.name("mainAddress")
               .getter(Person::getMainAddress)
               .setter(Person::setMainAddress))
           .addAttribute(EnhancedType.listOf(String.class), a -> a.name("hobbies")
               .getter(Person::getHobbies)
               .setter(Person::setHobbies))
           .addAttribute(EnhancedType.mapOf(
               EnhancedType.of(String.class),
               // 1. Use mapping functionality of the Address table schema.
```

```
                EnhancedType.documentOf(Address.class, TABLE_SCHEMA_ADDRESS)), a ->
a.name("addresses")
                .getter(Person::getAddresses)
                .setter(Person::setAddresses))
            .addAttribute(EnhancedType.listOf(
                // 2. Use mapping functionality of the PhoneNumber table schema.
                EnhancedType.documentOf(PhoneNumber.class, TABLE_SCHEMA_PHONENUMBER)),
a -> a.name("phoneNumbers")
                .getter(Person::getPhoneNumbers)
                .setter(Person::setPhoneNumbers))
            .build();
```

复杂类型的项目属性

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> hobbiesScanResult = personDynamoDbTable.scan(b -> b
                // Use the 'attributesToProject()' method to access first-level
 attributes.
                .attributesToProject("hobbies"));

        hobbiesScanResult.items().forEach((p) -> {
            // 3. Log the results of the request for the 'hobbies' attribute.
            logger.info(p.toString());
            // To access an item in a list, first get the parent attribute, 'hobbies',
 then access items in the list.
            String hobby = p.getHobbies().get(1);
            // 4. Log an item in the list.
            logger.info(hobby);
        });
```

```
// Logged results from comment line 1.
Person{id=2, firstName='first name 2', lastName='last name 2', age=11,
 addresses={work=Address{street='street 21', city='city 21', state='state 21',
 zipCode='33333'}, home=Address{street='street 2', city='city 2', state='state 2',
 zipCode='22222'}}, phoneNumbers=[PhoneNumber{type='home', number='222-222-2222'},
 PhoneNumber{type='work', number='333-333-3333'}], hobbies=[hobby 2, hobby 21]}
Person{id=1, firstName='first name 1', lastName='last name 1', age=11,
 addresses={work=Address{street='street 11', city='city 11', state='state 11',
 zipCode='22222'}, home=Address{street='street 1', city='city 1', state='state 1',
 zipCode='11111'}}, phoneNumbers=[PhoneNumber{type='home', number='111-111-1111'},
 PhoneNumber{type='work', number='222-222-2222'}], hobbies=[hobby 1, hobby 11]}

// Logged results from comment line 2.
```

```
Person{id=null, firstName='null', lastName='null', age=null,
 addresses={work=Address{street='street 21', city='null', state='null',
 zipCode='null'}}, phoneNumbers=null, hobbies=null}
Person{id=null, firstName='null', lastName='null', age=null,
 addresses={work=Address{street='street 11', city='null', state='null',
 zipCode='null'}}, phoneNumbers=null, hobbies=null}

// Logged results from comment lines 3 and 4.
Person{id=null, firstName='null', lastName='null', age=null, addresses=null,
 phoneNumbers=null, hobbies=[hobby 2, hobby 21]}
hobby 21
Person{id=null, firstName='null', lastName='null', age=null, addresses=null,
 phoneNumbers=null, hobbies=[hobby 1, hobby 11]}
hobby 11
```

Note

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]}
```

在表达式中使用复杂类型

您可以在表达式（例如筛选表达式和条件表达式）中使用复杂类型，通过使用解除引用操作符来访问复杂类型的结构。对于对象和 map，使用 `.`（dot）；对于 list 元素，使用 `[n]`（元素序号加上方括号）。不能直接引用 set 中的单个元素，但可以使用 `contains` 函数。

以下示例显示了在扫描操作中使用的两个筛选表达式。筛选表达式为要在结果中显示的项目指定匹配条件。该示例使用前面显示的 `Person`、`Address` 和 `PhoneNumber` 类。

```
   public void scanUsingFilterOfNestedAttr() {
       // The following is a filter expression for an attribute that is a map of
Address objects.
       // By using this filter expression, the SDK returns Person objects that have an
address
       // with 'mailing' as a key and 'MS2' for a state value.
       Expression addressFilter = Expression.builder()
               .expression("addresses.#type.#field = :value")
               .putExpressionName("#type", "mailing")
```

```
                .putExpressionName("#field", "state")
                .putExpressionValue(":value",
AttributeValue.builder().s("MS2").build())
                .build();

        PageIterable<Person> addressFilterResults = personDynamoDbTable.scan(rb -> rb.
                filterExpression(addressFilter));
        addressFilterResults.items().stream().forEach(p -> logger.info("Person: {}",
p));

        assert addressFilterResults.items().stream().count() == 1;


        // The following is a filter expression for an attribute that is a list of
phone numbers.
        // By using this filter expression, the SDK returns Person objects whose second
phone number
        // in the list has a type equal to 'cell'.
        Expression phoneFilter = Expression.builder()
                .expression("phoneNumbers[1].#type = :type")
                .putExpressionName("#type", "type")
                .putExpressionValue(":type",
AttributeValue.builder().s("cell").build())
                .build();

        PageIterable<Person> phoneFilterResults = personDynamoDbTable.scan(rb -> rb
                .filterExpression(phoneFilter)
                .attributesToProject("id", "firstName", "lastName", "phoneNumbers")
        );

        phoneFilterResults.items().stream().forEach(p -> logger.info("Person: {}", p));

        assert phoneFilterResults.items().stream().count() == 1;
        assert
phoneFilterResults.items().stream().findFirst().get().getPhoneNumbers().get(1).getType().equal
    }
```

用于向表中填充数据的辅助方法

```
    public static void populateDatabase() {
        Person person1 = new Person();
        person1.setId(1);
        person1.setFirstName("FirstName1");
```

```
        person1.setLastName("LastName1");

        Address billingAddr1 = new Address();
        billingAddr1.setState("BS1");
        billingAddr1.setCity("BillingTown1");

        Address mailing1 = new Address();
        mailing1.setState("MS1");
        mailing1.setCity("MailingTown1");

        person1.setAddresses(Map.of("billing", billingAddr1, "mailing", mailing1));

        PhoneNumber pn1_1 = new PhoneNumber();
        pn1_1.setType("work");
        pn1_1.setNumber("111-111-1111");

        PhoneNumber pn1_2 = new PhoneNumber();
        pn1_2.setType("home");
        pn1_2.setNumber("222-222-2222");

        List<PhoneNumber> phoneNumbers1 = List.of(pn1_1, pn1_2);
        person1.setPhoneNumbers(phoneNumbers1);

        personDynamoDbTable.putItem(person1);

        Person person2 = person1;
        person2.setId(2);
        person2.setFirstName("FirstName2");
        person2.setLastName("LastName2");

        Address billingAddress2 = billingAddr1;
        billingAddress2.setCity("BillingTown2");
        billingAddress2.setState("BS2");

        Address mailing2 = mailing1;
        mailing2.setCity("MailingTown2");
        mailing2.setState("MS2");

        person2.setAddresses(Map.of("billing", billingAddress2, "mailing", mailing2));

        PhoneNumber pn2_1 = new PhoneNumber();
        pn2_1.setType("work");
        pn2_1.setNumber("333-333-3333");
```

```
        PhoneNumber pn2_2 = new PhoneNumber();
        pn2_2.setType("cell");
        pn2_2.setNumber("444-444-4444");

        List<PhoneNumber> phoneNumbers2 = List.of(pn2_1, pn2_2);
        person2.setPhoneNumbers(phoneNumbers2);

        personDynamoDbTable.putItem(person2);
    }
```

数据库中项目的 JSON 表示形式

```
{
 "id": 1,
 "addresses": {
  "billing": {
   "city": "BillingTown1",
   "state": "BS1",
   "street": null,
   "zipCode": null
  },
  "mailing": {
   "city": "MailingTown1",
   "state": "MS1",
   "street": null,
   "zipCode": null
  }
 },
 "firstName": "FirstName1",
 "lastName": "LastName1",
 "phoneNumbers": [
  {
   "number": "111-111-1111",
   "type": "work"
  },
  {
   "number": "222-222-2222",
   "type": "home"
  }
 ]
}

{
```

```
 "id": 2,
 "addresses": {
  "billing": {
   "city": "BillingTown2",
   "state": "BS2",
   "street": null,
   "zipCode": null
  },
  "mailing": {
   "city": "MailingTown2",
   "state": "MS2",
   "street": null,
   "zipCode": null
  }
 },
 "firstName": "FirstName2",
 "lastName": "LastName2",
 "phoneNumbers": [
  {
   "number": "333-333-3333",
   "type": "work"
  },
  {
   "number": "444-444-4444",
   "type": "cell"
  }
 ]
}
```

更新包含复杂类型的项目

要更新包含复杂类型的项目，有两种基本方法：

- 方法 1：先检索项目（使用 `getItem`），更新对象，然后调用 `DynamoDbTable#updateItem`。
- 方法 2：不检索项目，而是构造一个新实例，设置要更新的属性，然后通过设置适当的 `IgnoreNullsMode` 值将实例提交到 `DynamoDbTable#updateItem`。这种方法不需要在更新之前获取项目。

此部分中显示的示例使用前面显示的 `Person`、`Address` 和 `PhoneNumber` 类。

更新方法 1：检索，然后更新

通过使用这种方法，您可以确保更新时不会丢失任何数据。DynamoDB 增强型客户端 API 使用保存在 DynamoDB 中的项目中的属性（包括复杂类型的值）重新创建 bean。然后，您需要使用 getter 和 setter 来更新 bean。这种方法的缺点是先检索项目会产生费用。

下面的示例说明：如果在更新项目之前先检索项目，则不会丢失任何数据。

```
    public void retrieveThenUpdateExample()  {
        // Assume that we ran this code yesterday.
        Person person = new Person();
        person.setId(1);
        person.setFirstName("FirstName");
        person.setLastName("LastName");

        Address mainAddress = new Address();
        mainAddress.setStreet("123 MyStreet");
        mainAddress.setCity("MyCity");
        mainAddress.setState("MyState");
        mainAddress.setZipCode("MyZipCode");
        person.setMainAddress(mainAddress);

        PhoneNumber homePhone = new PhoneNumber();
        homePhone.setNumber("1111111");
        homePhone.setType("HOME");
        person.setPhoneNumbers(List.of(homePhone));

        personDynamoDbTable.putItem(person);

        // Assume that we are running this code now.
        // First, retrieve the item
        Person retrievedPerson =
personDynamoDbTable.getItem(Key.builder().partitionValue(1).build());

        // Make any updates.
        retrievedPerson.getMainAddress().setCity("YourCity");

        // Save the updated bean. 'updateItem' returns the bean as it appears after the
update.
        Person updatedPerson = personDynamoDbTable.updateItem(retrievedPerson);

        // Verify for this example.
        Address updatedMainAddress = updatedPerson.getMainAddress();
```

```
        assert updatedMainAddress.getCity().equals("YourCity");
        assert updatedMainAddress.getState().equals("MyState"); // Unchanged.
        // The list of phone numbers remains; it was not set to null;
        assert updatedPerson.getPhoneNumbers().size() == 1;
    }
```

更新方法 2：使用 **IgnoreNullsMode** 枚举而不先检索项目

要更新 DynamoDB 中的项目，您可以提供一个仅包含待更新属性的新对象，而将其他值保留为 null。使用这种方法，您需要了解 SDK 如何处理对象中的 null 值以及如何控制行为。

要指定希望 SDK 忽略哪些 null 值属性，请在构建 UpdateItemEnhancedRequest 时提供 IgnoreNullsMode 枚举。作为枚举值使用的一个示例，以下代码段使用了 IgnoreNullsMode.SCALAR_ONLY 模式。

```
// Create a new Person object to update the existing item in DynamoDB.
Person personForUpdate = new Person();
personForUpdate.setId(1);
personForUpdate.setFirstName("updatedFirstName");  // 'firstName' is a top scalar
 property.

Address addressForUpdate = new Address();
addressForUpdate.setCity("updatedCity");
personForUpdate.setMainAddress(addressForUpdate);

personDynamoDbTable.updateItem(r -> r
                .item(personForUpdate)
                .ignoreNullsMode(IgnoreNullsMode.SCALAR_ONLY));

/* With IgnoreNullsMode.SCALAR_ONLY provided, The SDK ignores all null properties. The
 SDK adds or replaces
the 'firstName' property with the provided value, "updatedFirstName". The SDK updates
 the 'city' value of
'mainAddress', as long as the 'mainAddress' attribute already exists in DynamoDB.

In the background, the SDK generates an update expression that it sends in the request
 to DynamoDB.
The following JSON object is a simplified version of what it sends. Notice that the SDK
 includes the paths
to 'mainAddress.city' and 'firstName' in the SET clause of the update expression. No
 null values in
'personForUpdate' are included.
```

```
{
  "TableName": "PersonTable",
  "Key": {
    "id": {
      "N": "1"
    }
  },
  "ReturnValues": "ALL_NEW",
  "UpdateExpression": "SET #mainAddress.#city = :mainAddress_city, #firstName
 = :firstName",
  "ExpressionAttributeNames": {
    "#city": "city",
    "#firstName": "firstName",
    "#mainAddress": "mainAddress"
  },
  "ExpressionAttributeValues": {
    ":firstName": {
      "S": "updatedFirstName"
    },
    ":mainAddress_city": {
      "S": "updatedCity"
    }
  }
}

Had we chosen 'IgnoreNullsMode.DEFAULT' instead of 'IgnoreNullsMode.SCALAR_ONLY', the
 SDK would have included
null values in the "ExpressionAttributeValues" section of the request as shown in the
 following snippet.

  "ExpressionAttributeValues": {
    ":mainAddress": {
      "M": {
        "zipCode": {
          "NULL": true
        },
        "city": {
          "S": "updatedCity"
        },
        "street": {
          "NULL": true
        },
        "state": {
          "NULL": true
```

```
        }
      }
    },
    ":firstName": {
      "S": "updatedFirstName"
    }
  }
*/
```

《Amazon DynamoDB 开发人员指南》包含了有关更新表达式的更多信息。

`IgnoreNullsMode` 选项的描述

- `IgnoreNullsMode.SCALAR_ONLY` - 使用此设置在任何级别更新标量属性。SDK 会构造一个更新语句，该语句仅向 DynamoDB 发送非 null 的标量属性。SDK 会忽略 bean 或 map 的 null 值标量属性，保留在 DynamoDB 中保存的值。

  更新 map 或 bean 的标量属性时，map 必须已存在于 DynamoDB 中。如果 DynamoDB 中的对象尚不存在您向对象添加的 map 或 bean，则您会收到 `DynamoDbException`，其中包含消息：在更新表达式中提供的文档路径对于更新无效。您必须使用 `MAPS_ONLY` 模式，先将 bean 或 map 添加到 DynamoDB 中，然后才能更新其属性。
- `IgnoreNullsMode.MAPS_ONLY` - 使用此设置添加或替换 bean 或 map 的属性。SDK 会替换或添加对象中提供的任何 map 或 bean。系统会忽略对象中为 null 的任何 bean 或 map，并保留 DynamoDB 中存在的 map。
- `IgnoreNullsMode.DEFAULT` - 使用此设置，SDK 永远不会忽略 null 值。任何级别的标量属性如果为 null，则更新为 null。在 DynamoDB 中，SDK 会将对象中的任何 null 值 bean、map、list 或 set 属性更新为 null。当您使用此模式（或者由于这是默认模式而不提供模式）时，应先检索项目，以便 DynamoDB 中的值不会设置为更新对象中提供的 null，除非您确实有意将这些值设置为 null。

在所有模式下，如果您将具有非 null 值 list 或 set 属性的对象提供给 `updateItem`，则 list 或 set 会保存到 DynamoDB 中。

为什么使用这些模式？

当您向 `updateItem` 方法提供一个具有 bean 或 map 的对象时，SDK 无法判断是应该使用 bean 中的属性值（或 map 中的条目值）来更新项目，还是应该使用整个 bean/map 来替换保存到 DynamoDB 的内容。

基于我们之前“先检索项再更新”的示例，现在尝试在不检索的情况下直接更新 `mainAddress` 的 `city` 属性。

```
/* The retrieval example saved the Person object with a 'mainAddress' property whose
 'city' property value is "MyCity".
/* Note that we create a new Person with only the necessary information to update the
 city value
of the mainAddress. */
Person personForUpdate = new Person();
personForUpdate.setId(1);
// The update we want to make changes the city.
Address mainAddressForUpdate = new Address();
mainAddressForUpdate.setCity("YourCity");
personForUpdate.setMainAddress(mainAddressForUpdate);

// Lets' try the following:
Person updatedPerson = personDynamoDbTable.updateItem(personForUpdate);
/*
 Since we haven't retrieved the item, we don't know if the 'mainAddress' property
 already exists, so what update expression should the SDK generate?

A) Should it replace or add the 'mainAddress' with the provided object (setting all
 attributes to null other than city)
   as shown in the following simplified JSON?

      {
        "TableName": "PersonTable",
        "Key": {
          "id": {
            "N": "1"
          }
        },
        "ReturnValues": "ALL_NEW",
        "UpdateExpression": "SET #mainAddress = :mainAddress",
        "ExpressionAttributeNames": {
          "#mainAddress": "mainAddress"
        },
        "ExpressionAttributeValues": {
          ":mainAddress": {
            "M": {
              "zipCode": {
                "NULL": true
              },
              "city": {
                "S": "YourCity"
              },
```

```
              "street": {
                "NULL": true
              },
              "state": {
                "NULL": true
              }
            }
          }
        }
      }

B) Or should it update only the 'city' attribute of an existing 'mainAddress' as shown
 in the following simplified JSON?

      {
        "TableName": "PersonTable",
        "Key": {
          "id": {
            "N": "1"
          }
        },
        "ReturnValues": "ALL_NEW",
        "UpdateExpression": "SET #mainAddress.#city = :mainAddress_city",
        "ExpressionAttributeNames": {
          "#city": "city",
          "#mainAddress": "mainAddress"
        },
        "ExpressionAttributeValues": {
          ":mainAddress_city": {
            "S": "YourCity"
          }
        }
      }

However, assume that we don't know if the 'mainAddress' already exists. If it doesn't
 exist, the SDK would try to update
an attribute of a non-existent map, which results in an exception.

In this particular case, we would likely select option B (SCALAR_ONLY) to retain the
 other values of the 'mainAddress'.
*/
```

以下两个示例展示了 MAPS_ONLY 和 SCALAR_ONLY 枚举值的用法。MAPS_ONLY 添加 map，而 SCALAR_ONLY 更新 map。

IgnoreNullsMode.MAPS_ONLY 示例

```
    public void mapsOnlyModeExample() {
        // Assume that we ran this code yesterday.
        Person person = new Person();
        person.setId(1);
        person.setFirstName("FirstName");

        personDynamoDbTable.putItem(person);

        // Assume that we are running this code now.

        /* Note that we create a new Person with only the necessary information to
update the city value
        of the mainAddress. */
        Person personForUpdate = new Person();
        personForUpdate.setId(1);
        // The update we want to make changes the city.
        Address mainAddressForUpdate = new Address();
        mainAddressForUpdate.setCity("YourCity");
        personForUpdate.setMainAddress(mainAddressForUpdate);


        Person updatedPerson = personDynamoDbTable.updateItem(r -> r
                .item(personForUpdate)
                .ignoreNullsMode(IgnoreNullsMode.MAPS_ONLY)); // Since the mainAddress
property does not exist, use MAPS_ONLY mode.
        assert updatedPerson.getMainAddress().getCity().equals("YourCity");
        assert updatedPerson.getMainAddress().getState() == null;
    }
```

IgnoreNullsMode.SCALAR_ONLY example

```
    public void scalarOnlyExample() {
        // Assume that we ran this code yesterday.
        Person person = new Person();
        person.setId(1);
        Address mainAddress = new Address();
        mainAddress.setCity("MyCity");
        mainAddress.setState("MyState");
```

```
        person.setMainAddress(mainAddress);

        personDynamoDbTable.putItem(person);

        // Assume that we are running this code now.

        /* Note that we create a new Person with only the necessary information to
update the city value
        of the mainAddress. */
        Person personForUpdate = new Person();
        personForUpdate.setId(1);
        // The update we want to make changes the city.
        Address mainAddressForUpdate = new Address();
        mainAddressForUpdate.setCity("YourCity");
        personForUpdate.setMainAddress(mainAddressForUpdate);

        Person updatedPerson = personDynamoDbTable.updateItem(r -> r
                .item(personForUpdate)
                .ignoreNullsMode(IgnoreNullsMode.SCALAR_ONLY)); // SCALAR_ONLY mode
ignores null properties in the in mainAddress.
        assert updatedPerson.getMainAddress().getCity().equals("YourCity");
        assert updatedPerson.getMainAddress().getState().equals("MyState"); // The
state property remains the same.
    }
```

请参阅下表，了解每种模式忽略哪些 null 值。除非您在处理 bean 或 map，否则通常可以使用 SCALAR_ONLY 或 MAPS_ONLY 中的任意一种。

对于每种模式，SDK 会忽略提交给 **`updateItem`** 的对象中的哪些 null 值属性？

属性的类型	在 SCALAR_ONLY 模式下	在 MAPS_ONLY 模式下	在 DEFAULT 模式下
顶层标量	支持	是	否
bean 或 map	支持	是	否
bean 或 map 条目的标量值	是[1]	否[2]	否
list 或 set	支持	是	否

[1]假设 map 已存在于 DynamoDB 中。您在更新对象中提供的 bean 或 map 的任何标量值（null 或非 null）都要求 DynamoDB 中存在该值的路径。SDK 在提交请求之前会使用 `.`（dot）解除引用操作符来构造指向该属性的路径。

[2]由于您使用 `MAPS_ONLY` 模式来完全替换或添加 bean 或 map，因此 bean 或 map 中的所有 null 值都将在保存到 DynamoDB 的 map 中保留。

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。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 getAddressesForCustomMappingExample() helper method that provides
'addresses' shows the use of a custom Address class
        // rather than using a Map<String, Map<String, String> to hold the address
data.
        Map<String, Address> addresses = getAddressesForCustomMappingExample();

        // Build an EnhancedDocument instance to save an item with a mix of structures
defined as needed and static classes.
        EnhancedDocument personDocument = EnhancedDocument.builder()
                .putNumber("id", 50)
                .putString("firstName", "Shirley")
                .putString("lastName", "Rodriguez")
                .putNumber("age", 53)
                .putNull("nullAttribute")
                .putJson("phoneNumbers", phoneNumbersJSONString())
                // Note the use of 'EnhancedType.of(Address.class)' instead of the more
generic
                // 'EnhancedType.mapOf(EnhancedType.of(String.class),
EnhancedType.of(String.class))' that was used in a previous example.
                .putMap("addresses", addresses, EnhancedType.of(String.class),
EnhancedType.of(Address.class))
                .putList("hobbies", List.of("Hobby 1", "Hobby 2"),
EnhancedType.of(String.class))
                .build();
        // Save the item to DynamoDB.
        documentDynamoDbTable.putItem(personDocument);

        // Retrieve the item just saved.
        EnhancedDocument srPerson =
documentDynamoDbTable.getItem(Key.builder().partitionValue(50).sortValue("Rodriguez").build())
```

```
        // Access the addresses attribute.
        Map<String, Address> srAddresses = srPerson.get("addresses",
                EnhancedType.mapOf(EnhancedType.of(String.class),
 EnhancedType.of(Address.class)));

        srAddresses.keySet().forEach(k -> logger.info(addresses.get(k).toString()));

        documentDynamoDbTable.deleteTable();

// The content logged to the console shows that the saved maps were converted to
 Address instances.
Address{street='123 Main Street', city='Any Town', state='NC', zipCode='00000'}
Address{street='100 Any Street', city='Any Town', state='NC', zipCode='00000'}
```

`CustomAttributeConverterProvider` 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 oject.
        @Override
        public Address transformTo(AttributeValue attributeValue) {
            Map<String, AttributeValue> m = attributeValue.m();
            Address address = new Address();
            address.setStreet(m.get("street").s());
            address.setCity(m.get("city").s());
            address.setState(m.get("state").s());
            address.setZipCode(m.get("zipCode").s());

            return address;
        }

        @Override
        public EnhancedType<Address> type() {
            return EnhancedType.of(Address.class);
        }

        @Override
        public AttributeValueType attributeValueType() {
            return AttributeValueType.M;
        }
    }
}
```

`Address` 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。

使用扩展自定义 DynamoDB 增强型客户端操作

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。扩展使用两种钩子方法在读取和写入操作期间修改数据：

- `beforeWrite()` - 在写入操作实际执行之前对其进行修改
- `afterRead()` - 在读取操作完成之后对返回的结果进行修改

某些操作（例如项目更新）同时执行写入和读取，因此会调用两种钩子方法。

如何加载扩展

按照您在增强型客户端生成器中指定的顺序加载扩展。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。

默认情况下，增强版客户端会加载两个扩展：

- VersionedRecordExtension - 提供乐观锁
- AtomicCounterExtension - 自动增加计数器属性

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。

> **Important**
>
> 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。

以下示例说明如何在 VersionedRecordExtension 之后加载名为 verifyChecksumExtension 的自定义扩展。本示例中未加载 AtomicCounterExtension。

```
DynamoDbEnhancedClientExtension versionedRecordExtension =
 VersionedRecordExtension.builder().build();

DynamoDbEnhancedClient enhancedClient =
    DynamoDbEnhancedClient.builder()
                          .dynamoDbClient(dynamoDbClient)
                          .extensions(versionedRecordExtension,
 verifyChecksumExtension)
                          .build();
```

可用的扩展详细信息和配置

以下部分提供有关 SDK 中每个可用扩展的详细信息。

使用 **VersionedRecordExtension** 实施乐观锁

VersionedRecordExtension 扩展提供乐观锁，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。

配置

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())
```

工作方式

使用 VersionedRecordExtension 实现的乐观锁机制会对这些 DynamoDbEnhancedClient 和 DynamoDbTable 方法产生以下影响：

putItem

为新项目分配的初始版本值为 0。可以使用 @DynamoDbVersionAttribute(startAt = *X*) 进行配置。

updateItem

如果您检索项目，然后更新它的一个或多个属性，并尝试保存所做更改，那么只有在客户端和服务器端的版本号匹配时操作才能成功。

如果成功，版本号自动加 1。可以使用 @DynamoDbVersionAttribute(incrementBy = *X*) 进行配置。

deleteItem

DynamoDbVersionAttribute 注释没有效果。删除项目时，必须手动添加条件表达式。

以下示例添加了一个条件表达式，以确保删除的项目是读取的项目。在以下示例中，recordVersion 是 bean 使用 @DynamoDbVersionAttribute 进行注释的属性。

```
// 1. Read the item and get its current version.
Customer item =
 customerTable.getItem(Key.builder().partitionValue("someId").build());
```

```
// `recordVersion` is the bean's attribute that is annotated with
 `@DynamoDbVersionAttribute`.
AttributeValue currentVersion = item.getRecordVersion();

// 2. Create conditional delete with the `currentVersion` value.
DeleteItemEnhancedRequest deleteItemRequest =
    DeleteItemEnhancedRequest.builder()
       .key(KEY)
       .conditionExpression(Expression.builder()
           .expression("recordVersion = :current_version_value")
           .putExpressionValue(":current_version_value", currentVersion)
           .build()).build();

customerTable.deleteItem(deleteItemRequest);
```

`transactWriteItems`

- `addPutItem`：此方法的行为与 `putItem` 相同。
- `addUpdateItem`：此方法的行为与 `updateItem` 相同。
- `addDeleteItem`：此方法的行为与 `deleteItem` 相同。

`batchWriteItem`

- `addPutItem`：此方法的行为与 `putItem` 相同。
- `addDeleteItem`：此方法的行为与 `deleteItem` 相同。

> **Note**
>
> DynamoDB 全局表在处理并发更新时，使用'最后一个写入方为准'协调策略，DynamoDB 会尽最大努力判断谁是最后一个写入方。如果您使用全局表，这个"最后一个写入方为准"策略意味着锁定策略可能无法按预期工作，因为所有副本最终会基于 DynamoDB 确定的最后一次写入结果达成一致。

如何禁用

要禁用乐观锁，请不要使用 `@DynamoDbVersionAttribute` 注释。

使用 **`AtomicCounterExtension`** 实施计数器

每次向数据库写入记录时，`AtomicCounterExtension` 扩展都会增加一个带标签的数字属性。您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。

配置

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))
```

使用 **AutoGeneratedTimestampRecordExtension** 添加时间戳

每次成功将项目写入数据库时，AutoGeneratedTimestampRecordExtension 扩展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。

配置

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())
```

使用 AutoGeneratedUuidExtension 生成 UUID

向数据库写入新记录时，AutoGeneratedUuidExtension 扩展为属性生成唯一的 UUID（通用唯一标识符）。使用 Java JDK UUID.randomUUID() 方法并应用于 java.lang.String 类型的属性。默认情况下不加载此扩展。

配置

在以下代码段中，uniqueId 属性是扩展行为的目标。

```
    @AutoGeneratedUuidExtension
    public String getUniqueId() {...}
    public void setUniqueId(String uniqueId) {...}
```

下面的代码段显示了等效的静态表架构方法。

```
     .addAttribute(String.class, a -> a.name("uniqueId")
                                       .getter(Customer::getUniqueId)
                                       .setter(Customer::setUniqueId)
                                       // Applying the 'autoGeneratedUuid' tag to the
attribute.

.tags(AutoGeneratedUuidExtension.AttributeTags.autoGeneratedUuidAttribute())
```

如果您希望扩展仅为 putItem 方法填充 UUID，而不为 updateItem 方法填充 UUID，请添加更新行为注释，如以下代码段所示。

```
    @AutoGeneratedUuidExtension
    @DynamoDbUpdateBehavior(UpdateBehavior.WRITE_IF_NOT_EXISTS)
    public String getUniqueId() {...}
    public void setUniqueId(String uniqueId) {...}
```

如果您使用静态表架构方法，请使用以下等效代码。

```
     .addAttribute(String.class, a -> a.name("uniqueId")
```

```
                                        .getter(Customer::getUniqueId)
                                        .setter(Customer::setUniqueId)
                                        // Applying the 'autoGeneratedUuid' tag to the
attribute.

.tags(AutoGeneratedUuidExtension.AttributeTags.autoGeneratedUuidAttribute(),

StaticAttributeTags.updateBehavior(UpdateBehavior.WRITE_IF_NOT_EXISTS))
```

自定义扩展示例

您可以通过实施 DynamoDbEnhancedClientExtension 接口来创建自定义扩展。以下自定义扩展类展示了 beforeWrite() 方法，如果数据库中的项目还没有 registrationDate 属性，则使用更新表达式来设置该属性。

```
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()
                                  .dynamoDbClient(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 SdkPublisher，而不是同步 DynamoDbEnhanceClient 为相同方法返回的 SdkIterable。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
DynamoDbAtomicCounter	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。
DynamoDbAttribute	属性	定义或重命名映射到 DynamoDB 表属性的 Bean 属性。	• 初步讨论。 • 入门部分，请参阅“注意”。 • 在查询方法示例的 MovieActor 类中。
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。
DynamoDbAutoGeneratedUuid	属性	向数据库写入新记录时，为属性生成唯一的 UUID（通用唯一标识符）。	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 `AttributeConverter` 与带注释的属性相关联。	初步讨论和示例。

注释名称	注释适用于[1]	作用	本指南中显示的位置
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。	• 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。	• 初步讨论。 • 在 ProductCatalog 类中使用。
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。	• 注释简介。 • 在 ProductCatalog 类中使用。 • 与 Lombok 结合使用。
DynamoDbPartitionKey	属性	将属性标记为 DynamoDb 表的主分区键（哈希键）。	• 第一次是在“入门”部分的 Customer 类上使用。 • 与 Lombok 结合使用。
DynamoDbPreserveEmptyObject	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。

注释名称	注释适用于[1]	作用	本指南中显示的位置
DynamoDbSecondaryPartitionKey	属性	将属性标记为全局二级属性的分区键。	• 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。
DynamoDbSecondarySortKey	属性	将属性标记为全局或本地二级索引的可选排序键。	• 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。
DynamoDbSortKey	属性	将属性标记为可选的主排序键（范围键）。	• “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	指定在“更新”操作（例如 UpdateItem）中更新此属性时的行为。	简介和示例。
DynamoDbVersionAttribute	属性	递增项目版本号。	介绍和讨论。

[1]您可以对 getter 或 setter 应用属性级注释，但不能同时应用两者。本指南显示了有关 getter 的注释。

[2]对于封装在 JavaBean 数据类中的值，通常使用术语 `property`。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”。）

使用 Amazon EC2

本部分提供使用AWS SDK for Java 2.x 对 Amazon EC2 进行编程的示例。

主题

- 管理 Amazon EC2 实例
- 使用 AWS 区域 区和可用区
- 在 Amazon EC2 中使用安全组
- 使用 Amazon EC2 实例元数据

管理 Amazon EC2 实例

创建实例

通过调用 Ec2Client 的`runInstances`方法创建一个新 Amazon EC2 实例，为其提供RunInstancesRequest包含要使用的亚马逊系统映像 (AMI) 和实例类型。

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 "";
    }
```

请参阅上的完整示例 GitHub。

启动实例

要启动 Amazon EC2 实例，请调用 Ec2Client `startInstances`的方法，为其提供StartInstancesRequest包含要启动的实例的 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 startInstance(Ec2Client ec2, String instanceId) {

        StartInstancesRequest request = StartInstancesRequest.builder()
                .instanceIds(instanceId)
                .build();

        ec2.startInstances(request);
        System.out.printf("Successfully started instance %s", instanceId);
    }
```

请参阅上的完整示例 GitHub。

停止实例

要停止 Amazon EC2 实例，请调用 Ec2Client `stopInstances`的方法，为其提供StopInstancesRequest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);
```

```
    }
```

请参阅上的完整示例 GitHub。

重启实例

要重启 Amazon EC2 实例，请调用 Ec2Client `rebootInstances`的方法，为其提供RebootInstancesRequest包含要重启的实例 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);
     }
  }
```

请参阅上的完整示例 GitHub。

描述实例

要列出您的实例，请创建DescribeInstancesRequest并调用 Ec2Client `describeInstances`的方法。它将返回一个DescribeInstancesResponse对象，您可以使用该对象列出您的账户和地区的 Amazon EC2 实例。

实例按预留进行分组。每个预留对应对启动实例的 `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。

请参阅上的完整示例 GitHub。

监控实例

您可以监控 Amazon EC2 实例的各个方面，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更多信息，请参阅 Linux 实例 Amazon EC2 用户指南 Amazon EC2中的监控。

要开始监控实例，您必须MonitorInstancesRequest使用要监控的实例的 ID 创建一个，并将其传递给 Ec2Client `monitorInstances`的方法。

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);
    }
```

请参阅上的完整示例 GitHub。

停止实例监控

要停止监控实例，请UnmonitorInstancesRequest使用要停止监控的实例 ID 创建一个，然后将其传递给 Ec2Client unmonitorInstances的方法。

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);
    }
```

请参阅上的完整示例 GitHub。

更多信息

- RunInstances在 Amazon EC2 API 参考中
- DescribeInstances在 Amazon EC2 API 参考中
- StartInstances在 Amazon EC2 API 参考中
- StopInstances在 Amazon EC2 API 参考中
- RebootInstances在 Amazon EC2 API 参考中
- MonitorInstances在 Amazon EC2 API 参考中
- UnmonitorInstances在 Amazon EC2 API 参考中

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.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
import java.util.concurrent.CompletableFuture;
```

代码

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeRegionsAndZones {
    public static void main(String[] args) {
        Ec2AsyncClient ec2AsyncClient = Ec2AsyncClient.builder()
            .region(Region.US_EAST_1)
            .build();

        try {
            CompletableFuture<Void> future =
 describeEC2RegionsAndZonesAsync(ec2AsyncClient);
            future.join(); // Wait for both async operations to complete.
        } catch (RuntimeException rte) {
```

```
            System.err.println("An exception occurred: " + (rte.getCause() != null ?
rte.getCause().getMessage() : rte.getMessage()));
        }
    }

    /**
     * Asynchronously describes the EC2 regions and availability zones.
     *
     * @param ec2AsyncClient the EC2 async client used to make the API calls
     * @return a {@link CompletableFuture} that completes when both the region and
availability zone descriptions are complete
     */
    public static CompletableFuture<Void>
describeEC2RegionsAndZonesAsync(Ec2AsyncClient ec2AsyncClient) {
        // Initiate the asynchronous request to describe regions
        CompletableFuture<DescribeRegionsResponse> regionsResponse =
ec2AsyncClient.describeRegions();

        // Handle the response or exception for regions
        CompletableFuture<DescribeRegionsResponse> regionsFuture =
regionsResponse.whenComplete((regionsResp, ex) -> {
            if (ex != null) {
                // Handle the exception by throwing a RuntimeException
                throw new RuntimeException("Failed to describe EC2 regions.", ex);
            } else if (regionsResp == null || regionsResp.regions().isEmpty()) {
                // Throw an exception if the response is null or the result is empty
                throw new RuntimeException("No EC2 regions found.");
            } else {
                // Process the response if no exception occurred and the result is not
empty
                regionsResp.regions().forEach(region -> {
                    System.out.printf(
                        "Found Region %s with endpoint %s%n",
                        region.regionName(),
                        region.endpoint());
                });
            }
        });

        CompletableFuture<DescribeAvailabilityZonesResponse> zonesResponse =
ec2AsyncClient.describeAvailabilityZones();
        CompletableFuture<DescribeAvailabilityZonesResponse> zonesFuture =
zonesResponse.whenComplete((zonesResp, ex) -> {
            if (ex != null) {
```

```
                throw new RuntimeException("Failed to describe EC2 availability
 zones.", ex);
            } else if (zonesResp == null || zonesResp.availabilityZones().isEmpty()) {
                throw new RuntimeException("No EC2 availability zones found.");
            } else {
                zonesResp.availabilityZones().forEach(zone -> {
                    System.out.printf(
                        "Found Availability Zone %s with status %s in region %s%n",
                        zone.zoneName(),
                        zone.state(),
                        zone.regionName()
                    );
                });
            }
        });

        return CompletableFuture.allOf(regionsFuture, zonesFuture);
    }
}
```

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.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
import java.util.concurrent.CompletableFuture;
```

代码

创建 Ec2Client。

```
        Ec2AsyncClient ec2AsyncClient = Ec2AsyncClient.builder()
            .region(Region.US_EAST_1)
```

```
            .build();
```

然后调用 describeAvailabilityZones () 并检索结果。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.DescribeRegionsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesResponse;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeRegionsAndZones {
    public static void main(String[] args) {
        Ec2AsyncClient ec2AsyncClient = Ec2AsyncClient.builder()
            .region(Region.US_EAST_1)
            .build();

        try {
            CompletableFuture<Void> future =
 describeEC2RegionsAndZonesAsync(ec2AsyncClient);
            future.join(); // Wait for both async operations to complete.
        } catch (RuntimeException rte) {
            System.err.println("An exception occurred: " + (rte.getCause() != null ?
 rte.getCause().getMessage() : rte.getMessage()));
        }
    }

    /**
     * Asynchronously describes the EC2 regions and availability zones.
     *
     * @param ec2AsyncClient the EC2 async client used to make the API calls
     * @return a {@link CompletableFuture} that completes when both the region and
 availability zone descriptions are complete
     */
    public static CompletableFuture<Void>
 describeEC2RegionsAndZonesAsync(Ec2AsyncClient ec2AsyncClient) {
```

```
        // Initiate the asynchronous request to describe regions
        CompletableFuture<DescribeRegionsResponse> regionsResponse =
ec2AsyncClient.describeRegions();

        // Handle the response or exception for regions
        CompletableFuture<DescribeRegionsResponse> regionsFuture =
regionsResponse.whenComplete((regionsResp, ex) -> {
            if (ex != null) {
                // Handle the exception by throwing a RuntimeException
                throw new RuntimeException("Failed to describe EC2 regions.", ex);
            } else if (regionsResp == null || regionsResp.regions().isEmpty()) {
                // Throw an exception if the response is null or the result is empty
                throw new RuntimeException("No EC2 regions found.");
            } else {
                // Process the response if no exception occurred and the result is not
empty
                regionsResp.regions().forEach(region -> {
                    System.out.printf(
                        "Found Region %s with endpoint %s%n",
                        region.regionName(),
                        region.endpoint());
                });
            }
        });

        CompletableFuture<DescribeAvailabilityZonesResponse> zonesResponse =
ec2AsyncClient.describeAvailabilityZones();
        CompletableFuture<DescribeAvailabilityZonesResponse> zonesFuture =
zonesResponse.whenComplete((zonesResp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to describe EC2 availability
zones.", ex);
            } else if (zonesResp == null || zonesResp.availabilityZones().isEmpty()) {
                throw new RuntimeException("No EC2 availability zones found.");
            } else {
                zonesResp.availabilityZones().forEach(zone -> {
                    System.out.printf(
                        "Found Availability Zone %s with status %s in region %s%n",
                        zone.zoneName(),
                        zone.state(),
                        zone.regionName()
                    );
                });
            }
```

```
        });

        return CompletableFuture.allOf(regionsFuture, zonesFuture);
    }
}
```

请参阅上的完整示例 GitHub。

描述账户

要列出有关您账户的 EC2相关信息，请调用 Ec2Client `describeAccountAttributes` 的方法。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。您可以遍历该列表来检索 `AccountAttribute` 对象。

您可以通过调用 `AccountAttribute` 对象的 `attributeValues` 方法来获取您账户的属性值。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.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.DescribeAccountAttributesResponse;
import java.util.concurrent.CompletableFuture;
```

代码

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.DescribeAccountAttributesResponse;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeAccount {
    public static void main(String[] args) {
        Ec2AsyncClient ec2AsyncClient = Ec2AsyncClient.builder()
```

```
            .region(Region.US_EAST_1)
            .build();

        try {
            CompletableFuture<DescribeAccountAttributesResponse> future =
describeEC2AccountAsync(ec2AsyncClient);
            future.join();
            System.out.println("EC2 Account attributes described successfully.");
        } catch (RuntimeException rte) {
            System.err.println("An exception occurred: " + (rte.getCause() != null ?
rte.getCause().getMessage() : rte.getMessage()));
        }
    }

    /**
     * Describes the EC2 account attributes asynchronously.
     *
     * @param ec2AsyncClient the EC2 asynchronous client to use for the operation
     * @return a {@link CompletableFuture} containing the {@link
DescribeAccountAttributesResponse} with the account attributes
     */
    public static CompletableFuture<DescribeAccountAttributesResponse>
describeEC2AccountAsync(Ec2AsyncClient ec2AsyncClient) {
        CompletableFuture<DescribeAccountAttributesResponse> response =
ec2AsyncClient.describeAccountAttributes();
        return response.whenComplete((accountResults, ex) -> {
            if (ex != null) {
                // Handle the exception by throwing a RuntimeException.
                throw new RuntimeException("Failed to describe EC2 account
attributes.", ex);
            } else if (accountResults == null ||
accountResults.accountAttributes().isEmpty()) {
                // Throw an exception if the response is null or no account attributes
are found.
                throw new RuntimeException("No account attributes found.");
            } else {
                // Process the response if no exception occurred.
                accountResults.accountAttributes().forEach(attribute -> {
                    System.out.println("\nThe name of the attribute is " +
attribute.attributeName());
                    attribute.attributeValues().forEach(
                        myValue -> System.out.println("The value of the attribute is "
+ myValue.attributeValue()));
                });
```

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

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 Amazon EC2 中使用安全组

创建安全组

要创建安全组，请使用包含密钥名称的 CreateSecurityGroupRequest 调用 Ec2Client 的 `createSecurityGroup` 方法。

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 GitHub 上的完整示例。

配置安全组

安全组可以控制对 Amazon EC2 实例的入站 (入口) 流量和出站 (出口) 流量。

要向安全组添加入口规则，请使用 Ec2Client 的 `authorizeSecurityGroupIngress` 方法，提供安全组的名称和您想要在 AuthorizeSecurityGroupIngressRequest 对象中分配给安全组的访问规则 (IpPermission)。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 AuthorizeSecurityGroupEgressRequest 中向 Ec2Client 的 `authorizeSecurityGroupEgress` 方法提供相似的数据。

请参阅 GitHub 上的完整示例。

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。该方法返回 DescribeSecurityGroupsResponse，使用它后，您可以通过调用其 `securityGroups` 方法（返回一个 SecurityGroup 对象的列表）来访问安全组的列表。

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 GitHub 上的完整示例。

删除安全组

要删除安全组，请调用 Ec2Client 的 `deleteSecurityGroup` 方法，将其传递给一个包含要删除安全组 ID 的 DeleteSecurityGroupRequest。

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 GitHub 上的完整示例。

更多信息

- 《Amazon EC2 用户指南（适用于 Linux 实例）》中的 Amazon EC2 安全组
- 《Amazon EC2 用户指南（适用于 Linux 实例）》中的为您的 Linux 实例授权入站流量
- 《Amazon EC2 API Reference》中的 CreateSecurityGroup
- 《Amazon EC2 API Reference》中的 DescribeSecurityGroups
- 《Amazon EC2 API Reference》中的 DeleteSecurityGroup
- 《Amazon EC2 API Reference》中的 AuthorizeSecurityGroupIngress

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 “配置 HTTP 客户端” 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端。

```
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。

与... 一起工作 IAM

本节提供使用 AWS SDK for Java 2.x 进行编程 AWS Identity and Access Management (IAM) 的示例。

AWS Identity and Access Management (IAM) 使您能够安全地控制用户对 AWS 服务和资源的访问权限。使用 IAM，您可以创建和管理 AWS 用户和群组，并使用权限来允许和拒绝他们访问 AWS 资源。要获得完整的指南 IAM，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。完整的示例代码可在上找到 GitHub。您可以在那里下载单个源文件，也可以将存储库复制到本地以获得所有示例，然后构建并运行这些示例。

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 `IamClient's createAccessKey` 方法。

> **Note**
>
> 由于 是一项全局服务，因此，您必须将区域设置为 AWS_GLOBALIamClient 才能使 IAM 调用生效。

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 GitHub 上的完整示例。

列出访问密钥

要列出指定用户的访问密钥，请创建一个 ListAccessKeysRequest 对象，其中包含要为其列出密钥的用户名，并将该对象传递给 IamClient's listAccessKeys 方法。

> **Note**
>
> 如果您未向 listAccessKeys 提供用户名，则它将尝试列出与签署该请求的 AWS 账户相关联的访问密钥。

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)。您可以调用返回的 ListAccessKeysResponse 对象中的 isTruncated，以查看该查询返回的结果是否少于可用结果。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 GitHub 上的完整示例。

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（可通过 GetAccessKeyLastUsedRequest 对象传入）调用 IamClient's getAccessKeyLastUsed 方法。

然后，您可以使用返回的 GetAccessKeyLastUsedResponse 对象检索上次使用密钥的时间。

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 GitHub 上的完整示例。

激活或停用访问密钥

您可以激活或停用访问密钥，方式是创建 `UpdateAccessKeyRequest` 对象，提供访问密钥 ID、用户名（可选）和所需 `status`，然后将请求对象传递给 `IamClient`'s `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 GitHub 上的完整示例。

删除访问密钥

要永久删除访问密钥，请调用 IamClient’s deleteKey 方法，并为该方法提供包含访问密钥 ID 和用户名的 DeleteAccessKeyRequest。

> **Note**
>
> 密钥在删除后无法再检索或使用。要临时停用密钥，使其可以稍后再次激活，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 GitHub 上的完整示例。

更多信息

- 《IAM API Reference》中的 CreateAccessKey
- 《IAM API Reference》中的 ListAccessKeys
- 《IAM API Reference》中的 GetAccessKeyLastUsed
- 《IAM API Reference》中的 UpdateAccessKey
- 《IAM API Reference》中的 DeleteAccessKey

管理 IAM 用户

创建用户

使用包含用户名的 CreateUserRequest 对象向 IamClient 的 `createUser` 方法提供用户名，从而创建新 IAM 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 GitHub 上的完整示例。

列出用户

要列出账户中的 IAM 用户，请创建新的 ListUsersRequest 并将其传递给 IamClient 的 `listUsers` 方法。您可以通过在返回的 usersListUsersResponse 对象上调用 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 GitHub 上的完整示例。

更新用户

要更新用户，请调用 IamClient 对象的 `updateUser` 方法，该方法采用 UpdateUserRequest 对象，您可以使用它更改用户的名称 或路径。

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.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 GitHub 上的完整示例。

删除用户

要删除用户，请使用 UpdateUserRequest 对象（其中设置了要删除的用户名）调用 IamClient 的 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 GitHub 上的完整示例。

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
- 《IAM API Reference》中的 CreateUser
- 《IAM API Reference》中的 ListUsers

- 《IAM API Reference》中的 UpdateUser
- 《IAM API Reference》中的 DeleteUser

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>2.27.21</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》。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "dynamodb:GetItem",
        "Resource": "*",
        "Condition": {
            "DateGreaterThan": {
                "aws:CurrentTime": "2020-04-01T00:00:00Z"
            },
            "DateLessThan": {
                "aws:CurrentTime": "2020-06-30T23:59:59Z"
            }
        }
    }
}
```

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。该政策包含两个条件。

```
    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》。

JSON

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

```
        "Effect": "Allow",
        "Action": [
            "dynamodb:GetItem",
            "dynamodb:BatchGetItem",
            "dynamodb:Query",
            "dynamodb:PutItem",
            "dynamodb:UpdateItem",
            "dynamodb:DeleteItem",
            "dynamodb:BatchWriteItem"
        ],
        "Resource": "arn:aws:dynamodb:*:*:table/table-name",
        "Condition": {
            "ForAllValues:StringEquals": {
                "dynamodb:Attributes": [
                    "column-name1",
                    "column-name2",
                    "column-name3"
                ]
            },
            "StringEqualsIfExists": {
                "dynamodb:Select": "SPECIFIC_ATTRIBUTES"
            }
        }
    }
}
```

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》。

JSON

```
{
  "Version":"2012-10-17",
  "Statement" : {
    "Effect" : "Deny",
    "Principal" : "*",
    "Action" : "s3:*",
    "Resource" : [ "arn:aws:s3:::BUCKETNAME/*", "arn:aws:s3:::BUCKETNAME" ],
    "Condition" : {
      "ArnNotEquals" : {
        "aws:PrincipalArn" : "arn:aws:iam::444455556666:user/user-name"
      }
    }
  }
}
```

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:::amzn-s3-demo-bucket/*")
                        .addCondition(b1 -> b1
                                .operator(IamConditionOperator.STRING_EQUALS)
```

```
                                .key("s3:x-amz-acl")
                                .value("bucket-owner-full-control")))
                .build();
        return policy.toJson(IamPolicyWriter.builder()
                .prettyPrint(true).build());
    }
```

JSON 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》。

JSON

```
{
  "Version":"2012-10-17",
  "Statement" : {
    "Effect" : "Allow",
    "Principal" : {
      "AWS" : "111122223333"
    },
    "Action" : "s3:PutObject",
    "Resource" : "arn:aws:s3:::amzn-s3-demo-bucket/*",
    "Condition" : {
      "StringEquals" : {
        "s3:x-amz-acl" : "bucket-owner-full-control"
      }
    }
  }
}
```

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 CreatePolicyRequest 中向 IamClient 的 `createPolicy` 方法提供策略名称和 JSON 格式的策略文档。

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 GitHub 上的完整示例。

获取策略

要检索现有策略，请调用 IamClient 的 `getPolicy` 方法，并在 GetPolicyRequest 对象中提供策略的 ARN。

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 GitHub 上的完整示例。

附加角色策略

您可以将策略附加到 IAM 角色，方式是调用 IamClient 的 `attachRolePolicy` 方法，并在 AttachRolePolicyRequest 中为其提供角色名称和策略 ARN。

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 GitHub 上的完整示例。

列出附加的角色策略

通过调用 IamClient 的 `listAttachedRolePolicies` 方法列出角色中附加的策略。这需要 ListAttachedRolePoliciesRequest 对象，它包含要列出策略的角色名称。

在返回的 ListAttachedRolePoliciesResponse 对象上调用 `getAttachedPolicies` 来获取所附加策略的列表。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 GitHub 上的完整示例。

分离角色策略

要从角色分离策略，请调用 IamClient 的 `detachRolePolicy` 方法，并在 DetachRolePolicyRequest 中为其提供角色名称和策略 ARN。

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 GitHub 上的完整示例。

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。
- 《IAM API Reference》中的 CreatePolicy
- 《IAM API Reference》中的 GetPolicy
- 《IAM API Reference》中的 AttachRolePolicy
- 《IAM API Reference》中的 ListAttachedRolePolicies
- 《IAM API Reference》中的 DetachRolePolicy

使用 IAM 服务器证书

要在 AWS 上启用网站或应用程序的 HTTPS 连接，需要 SSL/TLS 服务器证书。您可以使用 AWS Certificate Manager 提供的服务器证书或您从外部提供程序获得的服务器证书。

我们建议您使用 ACM 来预配置、管理和部署您的服务器证书。利用 ACM，您可以申请证书，将其部署到 AWS 资源，然后让 ACM 为您处理证书续订事宜。ACM 提供的证书是免费的。有关 ACM 的更多信息，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 的 `getServerCertificate` 方法检索服务器证书，将包含证书名称的 GetServerCertificateRequest 传递给它。

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 GitHub 上的完整示例。

列出服务器证书

要列出服务器证书，请使用 ListServerCertificatesRequest 调用 IamClient 的 `listServerCertificates` 方法。该方法返回 ListServerCertificatesResponse。

调用返回的 `ListServerCertificateResponse` 对象的 `serverCertificateMetadataList` 方法获取 ServerCertificateMetadata 对象的列表，您可以用它来获取关于每个证书的信息。

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 GitHub 上的完整示例。

更新服务器证书

您可以通过调用 IamClient 的 `updateServerCertificate` 方法更新服务器证书的名称或路径。这需要通过服务器证书的当前名称以及要使用的新名称或新路径来设置 UpdateServerCertificateRequest 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.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 GitHub 上的完整示例。

删除服务器证书

要删除服务器证书，请使用包含证书名称的 DeleteServerCertificateRequest 调用 IamClient 的 `deleteServerCertificate` 方法。

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 GitHub 上的完整示例。

更多信息

- 《IAM 用户指南》中的使用服务器证书
- 《IAM API Reference》中的 GetServerCertificate
- 《IAM API Reference》中的 ListServerCertificates
- 《IAM API Reference》中的 UpdateServerCertificate
- 《IAM API Reference》中的 DeleteServerCertificate
- AWS Certificate Manager《用户指南》

使用 Kinesis

本部分提供使用AWS SDK for Java 2.x 对 Amazon Kinesis 进行编程的示例。

有关 Kinesis 的更多信息，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

- 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>2.2.21</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

本部分提供使用AWS SDK for Java 2.x 对 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

可以通过创建 LambdaClient 对象并调用其 invoke 方法来调用 Lambda 函数。创建 InvokeRequest 对象以指定其他信息，例如函数名称和要传递给 Lambda 函数的有效载荷。函数名称显示为 arn:aws:lambda:us-east-1:123456789012:function:HelloFunction。可以通过查看 AWS 管理控制台中的函数来检索值。

要将有效载荷数据传递给函数，请创建包含信息的 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 GitHub 上的完整示例。

列出 Lambda 函数

构建一个 LambdaClient 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 方法以返回 FunctionConfiguration 对象的列表。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 GitHub 上的完整示例。

删除 Lambda 函数

构建一个 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 管理控制台中的函数来检索值。

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 GitHub 上的完整示例。

使用 Amazon S3

本节提供有关使用AWS SDK for Java 2.x 来处理 Amazon S3 的背景信息。本节对本指南的代码示例部分中介绍的 Amazon S3 Java v2 示例作出了补充。

AWS SDK for Java 2.x 中的 S3 客户端

AWS SDK for Java 2.x 提供了不同类型的 S3 客户端。下表显示了不同客户端的差异，可以帮助您确定哪种客户端更适合您的使用案例。

Amazon S3 客户端的不同类型

S3 客户端	简短描述	何时使用	限制/缺点
基于 AWS CRT 的 S3 客户端 接口：S3AsyncClient 生成器：S3CrtAsyncClientBuilder	• 提供与基于 Java 的 S3 异步客户端相同的异步 API 操作，但性能更高。 • 需要 `aws-crt` 依赖项。 • 支持自动并行传输（分段）。 请参阅the section called “使用高性能 S3 客户端”。	• 您的应用程序要传输大型对象（> 8MB），而您想要更大限度地提高性能。 • 您想上传内容长度未知的对象。 • 您需要增强的连接池和 DNS 负载平衡，从而提高吞吐量和性能。 • 您希望在网络故障发生时提高传输的可靠性。重试各个失败的分段，而无需从头开始重新启动传输。	• 与基于 Java 的 S3 客户端相比，支持的配置设置更少。 • 需要额外的依赖项。
基于 Java 的 S3 异步客户端，启用了分段功能 接口：S3AsyncClient 生成器：S3AsyncClientBuilder	• 提供异步 API。 • 如果在创建时启用分段时，则支持自动并行传输（分段）。 请参阅the section called “配置并行传输支持”。	• 您的应用程序要传输大型对象，而您想要提高性能。 • 您想上传内容长度未知的对象。 • 您希望在网络故障发生时提高传输的可靠性。重试各个失败的分段，而无需从头开始重新启动传输。 • 您需要基于 AWS CRT 的 S3 客户端	性能不如基于 AWS CRT 的 S3 客户端。

S3 客户端	简短描述	何时使用	限制/缺点
		没有提供的配置选项。	
基于 Java 的 S3 异步客户端，未启用分段功能 接口：S3AsyncClient 生成器：S3AsyncClientBuilder	• 提供异步 API。	• 您正在传输小于 8 MB 的对象。 • 您想要一个异步 API。	没有性能优化。
基于 Java 的 S3 同步客户端 接口：S3Client 生成器：S3ClientBuilder	• 提供同步 API。	• 您正在传输小于 8 MB 的对象。 • 您想要一个同步 API。	没有性能优化。

Note

从版本 2.18.x 开始，在包括端点覆盖时，AWS SDK for Java 2.x 使用虚拟托管式寻址。这适用于只要桶名称是有效 DNS 标签的所有情况。
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();
```

主题

- 使用AWS SDK for Java 2.x 将流上传到 Amazon S3
- 使用 Amazon S3 预签名 URL
- Amazon S3 的跨区域访问
- 使用校验和实现数据完整性保护
- 使用高性能 S3 客户端：基于 AWS CRT 的 S3 客户端
- 将基于 Java 的 S3 异步客户端配置为使用并行传输
- 使用 Amazon S3 Transfer Manager 传输文件和目录
- 处理 S3 事件通知

使用AWS SDK for Java 2.x 将流上传到 Amazon S3

当您通过 putObject 或 uploadPart 方法，使用流将内容上传到 S3 时，可以使用 RequestBody 工厂类作为同步 API 来提供流。对于异步 API，AsyncRequestBody 是等效的工厂类。

哪些方法可上传流？

对于同步 API，您可以使用 RequestBody 的以下工厂方法来提供流：

- fromInputStream(InputStream inputStream, long contentLength)

  fromContentProvider(ContentStreamProvider provider, long contentLength, String mimeType)
  - ContentStreamProvider 有 fromInputStream(InputStream inputStream) 工厂方法
- fromContentProvider(ContentStreamProvider provider, String mimeType)

对于异步 API，您可以使用 AsyncRequestBody 的以下工厂方法：

- fromInputStream(InputStream inputStream, Long contentLength, ExecutorService executor)
- fromInputStream(AsyncRequestBodyFromInputStreamConfiguration configuration)
  - 使用 AsyncRequestBodyFromInputStreamConfiguration.Builder 来提供流
- fromInputStream(Consumer<AsyncRequestBodyFromInputStreamConfiguration.Builder> configuration)
- forBlockingInputStream(Long contentLength)

- 生成的 BlockingInputStreamAsyncRequestBody 包含 `writeInputStream(InputStream inputStream)` 方法，您可以使用该方法来提供流

执行上传

如果您知道流的长度

从前面所示方法的签名中可以看出，大多数方法都接受内容长度参数。

如果您知道以字节为单位的内容长度，请提供确切值：

```
// Always provide the exact content length when it's available.
long contentLength = 1024; // Exact size in bytes.
s3Client.putObject(req -> req
    .bucket("amzn-s3-demo-bucket")
    .key("my-key"),
RequestBody.fromInputStream(inputStream, contentLength));
```

> **Warning**
>
> 当您从输入流上传时，如果指定的内容长度与实际字节数不匹配，则您可能会遇到以下情况：
>
> - 如果指定的长度太小，则将截断对象
> - 如果指定的长度太大，则上传失败或连接挂起

如果您不知道流的长度

使用同步 API

使用 `fromContentProvider(ContentStreamProvider provider, String mimeType)`：

```
public PutObjectResponse syncClient_stream_unknown_size(String bucketName, String key,
 InputStream inputStream) {

    S3Client s3Client = S3Client.create();

    RequestBody body =
 RequestBody.fromContentProvider(ContentStreamProvider.fromInputStream(inputStream),
 "text/plain");
```

```
    PutObjectResponse putObjectResponse = s3Client.putObject(b ->
 b.bucket(BUCKET_NAME).key(KEY_NAME), body);
    return putObjectResponse;
}
```

由于 SDK 会将整个流缓冲到内存中来计算内容长度，所以在处理大型流时，可能会遇到内存不足的问题。如果您需要使用同步客户端上传大型流，请考虑使用分段 API：

使用同步客户端 API 和分段 API 上传流

```
public static void uploadStreamToS3(String bucketName, String key, InputStream
 inputStream) {
    // Create S3 client
    S3Client s3Client = S3Client.create();
    try {
        // Step 1: Initiate the multipart upload
        CreateMultipartUploadRequest createMultipartUploadRequest =
 CreateMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

        CreateMultipartUploadResponse createResponse =
 s3Client.createMultipartUpload(createMultipartUploadRequest);
        String uploadId = createResponse.uploadId();
        System.out.println("Started multipart upload with ID: " + uploadId);

        // Step 2: Upload parts
        List<CompletedPart> completedParts = new ArrayList<>();
        int partNumber = 1;
        byte[] buffer = new byte[PART_SIZE];
        int bytesRead;

        try {
            while ((bytesRead = readFullyOrToEnd(inputStream, buffer)) > 0) {
                // Create request to upload a part
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                        .bucket(bucketName)
                        .key(key)
                        .uploadId(uploadId)
                        .partNumber(partNumber)
                        .build();
```

```
                // If we didn't read a full buffer, create a properly sized byte array
                RequestBody requestBody;
                if (bytesRead < PART_SIZE) {
                    byte[] lastPartBuffer = new byte[bytesRead];
                    System.arraycopy(buffer, 0, lastPartBuffer, 0, bytesRead);
                    requestBody = RequestBody.fromBytes(lastPartBuffer);
                } else {
                    requestBody = RequestBody.fromBytes(buffer);
                }

                // Upload the part and save the response's ETag
                UploadPartResponse uploadPartResponse =
s3Client.uploadPart(uploadPartRequest, requestBody);
                CompletedPart part = CompletedPart.builder()
                        .partNumber(partNumber)
                        .eTag(uploadPartResponse.eTag())
                        .build();
                completedParts.add(part);

                System.out.println("Uploaded part " + partNumber + " with size " +
bytesRead + " bytes");
                partNumber++;
            }

            // Step 3: Complete the multipart upload
            CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
                    .parts(completedParts)
                    .build();

            CompleteMultipartUploadRequest completeRequest =
CompleteMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .multipartUpload(completedMultipartUpload)
                    .build();

            CompleteMultipartUploadResponse completeResponse =
s3Client.completeMultipartUpload(completeRequest);
            System.out.println("Multipart upload completed. Object URL: " +
completeResponse.location());

        } catch (Exception e) {
```

```
            // If an error occurs, abort the multipart upload
            System.err.println("Error during multipart upload: " + e.getMessage());
            AbortMultipartUploadRequest abortRequest =
 AbortMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .build();
            s3Client.abortMultipartUpload(abortRequest);
            System.err.println("Multipart upload aborted");
        } finally {
            try {
                inputStream.close();
            } catch (IOException e) {
                System.err.println("Error closing input stream: " + e.getMessage());
            }
        }
    } finally {
        s3Client.close();
    }
}

/**
 * Reads from the input stream into the buffer, attempting to fill the buffer
 completely
 * or until the end of the stream is reached.
 *
 * @param inputStream the input stream to read from
 * @param buffer      the buffer to fill
 * @return the number of bytes read, or -1 if the end of the stream is reached before
 any bytes are read
 * @throws IOException if an I/O error occurs
 */
private static int readFullyOrToEnd(InputStream inputStream, byte[] buffer) throws
 IOException {
    int totalBytesRead = 0;
    int bytesRead;

    while (totalBytesRead < buffer.length) {
        bytesRead = inputStream.read(buffer, totalBytesRead, buffer.length -
 totalBytesRead);
        if (bytesRead == -1) {
            break; // End of stream
        }
```

```
        totalBytesRead += bytesRead;
    }

    return totalBytesRead > 0 ? totalBytesRead : -1;
}
```

> **Note**
>
> 对于大多数使用案例，建议为未知大小的流使用异步客户端 API。这种方法支持并行传输，并提供更简单的编程接口，因为如果流很大，SDK 在处理流时会将其分割成多个分段。
> 启用了分段功能的标准 S3 异步客户端和基于 AWS CRT 的 S3 客户端都实现了这种方法。我们将在下一节介绍此方法的示例。

使用异步 API

您可以将 contentLength 参数 null 提供给 fromInputStream(InputStream inputStream, Long contentLength, ExecutorService executor)

Example使用基于 AWS CRT 的异步客户端：

```
public PutObjectResponse crtClient_stream_unknown_size(String bucketName, String key,
 InputStream inputStream) {

    S3AsyncClient s3AsyncClient = S3AsyncClient.crtCreate();
    ExecutorService executor = Executors.newSingleThreadExecutor();
    AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream, null,
 executor);  // 'null' indicates that the

     // content length is unknown.
    CompletableFuture<PutObjectResponse> responseFuture =
            s3AsyncClient.putObject(r -> r.bucket(bucketName).key(key), body)
                    .exceptionally(e -> {
                        if (e != null){
                            logger.error(e.getMessage(), e);
                        }
                        return null;
                    });

    PutObjectResponse response = responseFuture.join(); // Wait for the response.
    executor.shutdown();
    return response;
```

```
}
```

Example使用启用了分段的标准异步客户端：

```
public PutObjectResponse asyncClient_multipart_stream_unknown_size(String bucketName,
 String key, InputStream inputStream) {

    S3AsyncClient s3AsyncClient =
 S3AsyncClient.builder().multipartEnabled(true).build();
    ExecutorService executor = Executors.newSingleThreadExecutor();
    AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream, null,
 executor); // 'null' indicates that the

    // content length is unknown.
    CompletableFuture<PutObjectResponse> responseFuture =
            s3AsyncClient.putObject(r -> r.bucket(bucketName).key(key), body)
                    .exceptionally(e -> {
                        if (e != null) {
                            logger.error(e.getMessage(), e);
                        }
                        return null;
                    });

    PutObjectResponse response = responseFuture.join(); // Wait for the response.
    executor.shutdown();
    return response;
}
```

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 **HttpURLConnection**（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 v11 起）

```
    /* 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 SDK for Java 中的 **SdkHttpClient**

```
    /* Use the AWS SDK for Java SdkHttpClient class to do the download. */
    public byte[] useSdkHttpClientToGet(String presignedUrlString) {

        ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream(); //
Capture the response body to a byte array.
        try {
            URL presignedUrl = new URL(presignedUrlString);
            SdkHttpRequest request = SdkHttpRequest.builder()
                    .method(SdkHttpMethod.GET)
                    .uri(presignedUrl.toURI())
                    .build();
```

```
            HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
                    .request(request)
                    .build();

            try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
                HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
                response.responseBody().ifPresentOrElse(
                        abortableInputStream -> {
                            try {
                                IoUtils.copy(abortableInputStream,
byteArrayOutputStream);
                            } catch (IOException e) {
                                throw new RuntimeException(e);
                            }
                        },
                        () -> logger.error("No response body."));

                logger.info("HTTP Response code is {}",
response.httpResponse().statusCode());
            }
        } catch (URISyntaxException | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }
```

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 **HttpURLConnection**（自 v1.1 起）

```
    /* 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 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>2.27.21</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。

使用校验和实现数据完整性保护

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 Amazon S3 校验和的更多信息，请参阅 Amazon Simple Storage Service 用户指南，包括支持的算法。

您可以灵活地选择最适合自己需求的算法，并让 SDK 计算校验和。或者，您也可以使用任一受支持的算法，提供预先计算好的校验和值。

> **Note**
>
> 从AWS SDK for Java 2.x 的版本 2.30.0 开始，SDK 通过自动计算上传文件的 CRC32 校验和来提供默认的完整性保护。如果您未提供预先计算的校验和值，或未指定 SDK 计算校验和时应使用的算法，则 SDK 会计算此校验和。
> 该 SDK 还提供了可在外部设置的数据完整性保护全局设置，您可以在 AWS SDK 和工具参考指南中查阅相关说明。

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

使用 `putObject` 方法上传对象并提供校验和算法时，SDK 会计算指定算法的校验和。

以下代码段展示了上传具有 SHA256 校验和的对象的请求。当 SDK 发送请求时，它会计算 SHA256 校验和并上传对象。Amazon S3 会计算校验和并将其与 SDK 提供的校验和进行对比，从而验证内容的完整性。然后 Amazon S3 将校验和与对象一起存储。

```
public void putObjectWithChecksum() {
        s3Client.putObject(b -> b
                .bucket(bucketName)
                .key(key)
                .checksumAlgorithm(ChecksumAlgorithm.SHA256),
            RequestBody.fromString("This is a test"));
}
```

如果您未在请求中提供校验和算法，则校验和行为将根据您使用的 SDK 版本而异，具体如下表所示。

未提供校验和算法时的校验和行为

Java SDK 版本	校验和行为
2.30.0 之前的版本	SDK 不会自动计算基于 CRC 的校验和，也不会在请求中提供该值。
2.30.0 或更高版本	SDK 使用 CRC32 算法计算校验和，并在请求中提供该值。Amazon S3 通过计算自己的 CRC32 校验和并将其与 SDK 提供的校验和进行对比，从而验证传输的完整性。如果校验和匹配，则校验和将与对象一起保存。

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 SHA256 校验和的请求。

```
    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)));
    }
```

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。

适用于 Java 2.x 的 SDK 提供了两个选项，用于在分段上传中使用校验和。第一个选项使用 `S3TransferManager`。

以下 Transfer Manager 示例指定了用于上传的 SHA1 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();
    }
```

如果您在使用传输管理器上传时未提供校验和算法，则 SDK 会根据 CRC32 算法自动计算校验和。SDK 会对所有版本的 SDK 执行此计算。

第二个选项使用 S3Client API（或 S3AsyncClient API）来执行分段上传。如果用此方法来指定校验和，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()));
    }
```

完整示例的代码和测试位于 GitHub 代码示例存储库中。

下载对象

使用 getObject 方法下载对象时，在以下情况下 SDK 会自动验证校验和：如果适用于 `GetObjectRequest` 的生成器的 `checksumMode` 方法设置为 `ChecksumMode.ENABLED`。

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。

```
public GetObjectResponse getObjectWithChecksum() {
    return s3Client.getObject(b -> b
                    .bucket(bucketName)
                    .key(key)
                    .checksumMode(ChecksumMode.ENABLED))
            .response();
}
```

> **Note**
>
> 如果上传对象时没有使用校验和，则不会进行验证。

其他校验和计算选项

> **Note**
>
> 为了验证已传输数据的数据完整性以及识别任何传输错误，我们建议用户为校验和计算选项保留 SDK 默认设置。默认情况下，SDK 会为许多 S3 操作（包括 `PutObject` 和 `GetObject`）添加这项重要检查。

但如果您希望在使用 Amazon S3 时仅进行最少的校验和验证，则可以通过更改默认配置设置来禁用许多检查。

除非必要，否则请禁用自动校验和计算

对于支持校验和的操作（例如 `PutObject` 和 `GetObject`），您可以禁用 SDK 的自动校验和计算。但某些 S3 操作需要校验和计算；您不能为这些操作禁用校验和计算。

SDK 为请求的有效载荷和响应的有效载荷的校验和计算提供了单独的设置。

下面的列表说明了可用于在不同范围内更大限度地减少校验和计算的设置。

- 所有应用程序范围 - 通过更改环境变量或共享 AWS `config` 和 `credentials` 文件的配置文件中的设置，所有应用程序都可以使用这些设置。除非在应用程序或服务客户端范围内已覆盖，否则这些设置会影响所有 AWS SDK 应用程序中的所有服务客户端。
  - 在配置文件中添加设置：

```
[default]
request_checksum_calculation = WHEN_REQUIRED
response_checksum_validation = WHEN_REQUIRED
```

  - 添加环境变量：

```
AWS_REQUEST_CHECKSUM_CALCULATION=WHEN_REQUIRED
AWS_RESPONSE_CHECKSUM_VALIDATION=WHEN_REQUIRED
```

- 当前应用程序范围 - 您可以将 Java 系统属性 `aws.requestChecksumCalculation` 设置为 `WHEN_REQUIRED`，以便限制校验和计算。响应的相应系统属性为 `aws.responseChecksumValidation`。

  除非在服务客户端创建期间已覆盖，否则这些设置会影响应用程序中的所有 SDK 服务客户端。

  在应用程序启动时设置系统属性：

```
import software.amazon.awssdk.core.SdkSystemSetting;
import software.amazon.awssdk.core.checksums.RequestChecksumCalculation;
import software.amazon.awssdk.core.checksums.ResponseChecksumValidation;
import software.amazon.awssdk.services.s3.S3Client;

class DemoClass {
    public static void main(String[] args) {


 System.setProperty(SdkSystemSetting.AWS_REQUEST_CHECKSUM_CALCULATION.property(), //
 Resolves to "aws.requestChecksumCalculation".
               "WHEN_REQUIRED");

 System.setProperty(SdkSystemSetting.AWS_RESPONSE_CHECKSUM_VALIDATION.property(), //
 Resolves to "aws.responseChecksumValidation".
               "WHEN_REQUIRED");

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

        // Use s3Client.
```

```
    }
}
```

- 单个 S3 服务客户端范围 - 您可以使用生成器方法配置单个 S3 服务客户端，使其仅计算最少必要数量的校验和：

```
import software.amazon.awssdk.core.checksums.RequestChecksumCalculation;
import software.amazon.awssdk.services.s3.S3Client;

public class RequiredChecksums {
    public static void main(String[] args) {
        S3Client s3 = S3Client.builder()
                .requestChecksumCalculation(RequestChecksumCalculation.WHEN_REQUIRED)
                .responseChecksumValidation(ResponseChecksumValidation.WHEN_REQUIRED)
                .build();

        // Use s3Client.
    }
// ...
}
```

使用 **LegacyMd5Plugin** 以简化 MD5 兼容性

随着版本 2.30.0 引入对 CRC32 校验和的支持，SDK 停止计算所需操作的 MD5 校验和。

如果您的 S3 操作仍需要传统的 MD5 校验和行为，则可以使用 SDK 版本 2.31.32 中发布的 `LegacyMd5Plugin`。

当您需要与依赖传统 MD5 校验和行为的应用程序保持兼容时，尤其是在使用与 S3 兼容的第三方存储提供程序 [例如与 S3A 文件系统连接器（Apache Spark、Iceberg）一起使用的存储提供程序] 时，`LegacyMd5Plugin` 特别有用。

要使用 `LegacyMd5Plugin`，请将其添加到 S3 客户端生成器中：

```
// For synchronous S3 client.
S3Client s3Client = S3Client.builder()
                            .addPlugin(LegacyMd5Plugin.create())
                            .build();

// For asynchronous S3 client.
S3AsyncClient asyncClient = S3AsyncClient.builder()
                                         .addPlugin(LegacyMd5Plugin.create())
```

```
                                        .build();
```

如果您希望仅为需要校验和的操作添加 MD5 校验和，同时跳过为那些“支持但非必需校验和”的操作添加 SDK 默认校验和，则可以将 `ClientBuilder` 选项 `requestChecksumCalculation` 和 `responseChecksumValidation` 设置为 `WHEN_REQUIRED`。这样就只会将 SDK 默认校验和添加到需要校验和的操作中：

```
// Use the `LegacyMd5Plugin` with `requestChecksumCalculation` and
 `responseChecksumValidation` set to WHEN_REQUIRED.
S3AsyncClient asyncClient = S3AsyncClient.builder()
                                        .addPlugin(LegacyMd5Plugin.create())

 .requestChecksumCalculation(RequestChecksumCalculation.WHEN_REQUIRED)

 .responseChecksumValidation(ResponseChecksumValidation.WHEN_REQUIRED)
                                        .build();
```

当使用与 S3 兼容的第三方存储系统时，此配置特别有用，这些存储系统可能不完全支持较新的校验和算法，但某些操作仍需要 MD5 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。

> **Important**
>
> 基于 AWS CRT 的 S3 客户端目前不支持客户端级别和请求级别的 SDK 指标收集。

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>2.27.21</version>
</dependency>
<dependency>
  <groupId>software.amazon.awssdk.crt</groupId>
  <artifactId>aws-crt</artifactId>
  <version>0.30.11</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();
```

上传未知大小的流

基于 AWS AWS CRT 的 S3 客户端的一个显著优势是能够高效处理大小未知的输入流。当您需要从无法事先确定总大小的来源上传数据时，这特别有用。

```
public PutObjectResponse crtClient_stream_unknown_size(String bucketName, String key,
 InputStream inputStream) {

    S3AsyncClient s3AsyncClient = S3AsyncClient.crtCreate();
    ExecutorService executor = Executors.newSingleThreadExecutor();
    AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream, null,
 executor);  // 'null' indicates that the

     // content length is unknown.
    CompletableFuture<PutObjectResponse> responseFuture =
            s3AsyncClient.putObject(r -> r.bucket(bucketName).key(key), body)
                    .exceptionally(e -> {
                        if (e != null){
                            logger.error(e.getMessage(), e);
                        }
                        return null;
                    });

    PutObjectResponse response = responseFuture.join(); // Wait for the response.
    executor.shutdown();
    return response;
}
```

此功能有助于避免传统上传中的常见问题，即不正确的内容长度规格会导致对象被截断或上传失败。

配置限制

基于 AWS CRT 的 S3 客户端和基于 Java 的 S3 异步客户端提供了类似的功能，但基于 AWS CRT 的 S3 客户端具有性能优势。但是，基于 AWS CRT 的 S3 客户端缺少基于 Java 的 S3 异步客户端所具有的配置设置。这些设置包括：

- 客户端级别配置：API 调用尝试超时、压缩执行拦截器、指标发布者、自定义执行属性、自定义高级选项、自定义计划执行器服务、自定义标头
- 请求级别配置：自定义签署人、凭证提供程序、API 调用尝试超时

有关配置差异的完整列表，请参阅 API 参考。

基于 Java 的 S3 异步客户端	基于 AWS CRT 的 S3 客户端
客户端级别配置 • ClientOverrideConfiguration.Builder	客户端级别配置 • S3CrtAsyncClientBuilder
请求级别配置 • RequestOverrideConfiguration.Builder • AwsRequestOverrideConfiguration.Builder	没有请求级别配置

将基于 Java 的 S3 异步客户端配置为使用并行传输

从版本 2.27.5 开始，基于 Java 的标准 S3 异步客户端支持自动并行传输（分段上传和下载）。在创建基于 Java 的 S3 异步客户端时，您可以配置对并行传输的支持。

本节介绍如何启用并行传输以及如何自定义配置。

创建 `S3AsyncClient` 的实例

当您不在生成器上调用任何 `multipart*` 方法的情况下创建 `S3AsyncClient` 实例时，不会启用并行传输。以下每条语句都创建一个基于 Java 的 S3 异步客户端，不支持分段上传和下载。

在没有分段支持的情况下创建

Example

```
import software.amazon.awssdk.auth.credentials.ProcessCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3AsyncClient;


S3AsyncClient s3Client = S3AsyncClient.create();

S3AsyncClient s3Client2 = S3AsyncClient.builder().build();

S3AsyncClient s3Client3 = S3AsyncClient.builder()
        .credentialsProvider(ProcessCredentialsProvider.builder().build())
```

```
        .region(Region.EU_NORTH_1)
        .build();
```

在有分段支持的情况下创建

要使用默认设置启用并行传输，请在生成器上调用 `multipartEnabled` 并传入 `true`，如下例所示。

Example

```
S3AsyncClient s3AsyncClient2 = S3AsyncClient.builder()
        .multipartEnabled(true)
        .build();
```

`thresholdInBytes` 和 `minimumPartSizeInBytes` 设置的默认值为 8 MiB。

如果您自定义分段设置，则会自动启用并行传输，如下所示。

Example

```
import software.amazon.awssdk.services.s3.S3AsyncClient;
import static software.amazon.awssdk.transfer.s3.SizeConstant.MB;


S3AsyncClient s3AsyncClient2 = S3AsyncClient.builder()
        .multipartConfiguration(b -> b
                .thresholdInBytes(16 * MB)
                .minimumPartSizeInBytes(10 * MB))
        .build();
```

上传未知大小的流

启用了分段功能的基于 Java 的 S3 异步客户端可以有效地处理事先不知道总大小的输入流：

```
public PutObjectResponse asyncClient_multipart_stream_unknown_size(String bucketName,
 String key, InputStream inputStream) {

    S3AsyncClient s3AsyncClient =
 S3AsyncClient.builder().multipartEnabled(true).build();
    ExecutorService executor = Executors.newSingleThreadExecutor();
    AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream, null,
 executor); // 'null' indicates that the

    // content length is unknown.
```

```
    CompletableFuture<PutObjectResponse> responseFuture =
            s3AsyncClient.putObject(r -> r.bucket(bucketName).key(key), body)
                    .exceptionally(e -> {
                        if (e != null) {
                            logger.error(e.getMessage(), e);
                        }
                        return null;
                    });

    PutObjectResponse response = responseFuture.join(); // Wait for the response.
    executor.shutdown();
    return response;
}
```

这种方法可以防止手动指定错误的内容长度时可能出现的问题，例如对象截断或上传失败。

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)。

在基于 AWS CRT 的 S3 客户端或启用了分段的基于 Java 的标准 S3 异步客户端基础上构建时，S3 Transfer Manager 可以利用分段上传 API 和字节范围提取等性能改进。

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

要在使用 S3 Transfer Manager 时获得增强的分段性能，请使用必要的依赖项配置构建文件。

Use the AWS CRT-based S3 client

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>bom</artifactId>
            <version>2.27.21[1]</version>
            <type>pom</type>
            <scope>import</scope>
```

```
        </dependency>
    </dependencies>
</dependencyManagement>
<dependencies>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>s3-transfer-manager</artifactId>
    </dependency>
    <dependency>
        <groupId>software.amazon.awssdk.crt</groupId>
        <artifactId>aws-crt</artifactId>
        <version>0.29.143[2]</version>
    </dependency>
</dependencies>
```

[1] 最新版本。[2]最新版本。

Use the Java-based S3 async client

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>bom</artifactId>
            <version>2.27.21[1]</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
<dependencies>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>s3-transfer-manager</artifactId>
    </dependency>
</dependencies>
```

[1] 最新版本。

创建 S3 Transfer Manager 的实例

要启用并行传输，必须传入基于 AWS CRT 的 S3 客户端或启用了分段功能的基于 Java 的 S3 异步客户端。以下示例演示如何使用自定义设置来配置 S3 Transfer Manager。

Use the AWS CRT-based S3 client

```
        S3AsyncClient s3AsyncClient = S3AsyncClient.crtBuilder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .targetThroughputInGbps(20.0)
                .minimumPartSizeInBytes(8 * MB)
                .build();

        S3TransferManager transferManager = S3TransferManager.builder()
                .s3Client(s3AsyncClient)
                .build();
```

Use the Java-based S3 async client

如果构建文件中未包含 `aws-crt` 依赖项，则 S3 Transfer Manager 将在适用于 Java 的 SDK 2.x 中使用的基于 Java 的标准 S3 异步客户端的基础上构建。

S3 客户端的自定义配置 - 需要启用分段功能

```
        S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
                .multipartEnabled(true)
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        S3TransferManager transferManager = S3TransferManager.builder()
                .s3Client(s3AsyncClient)
                .build();
```

没有 S3 客户端配置 - 自动启用分段支持

```
S3TransferManager transferManager = S3TransferManager.create();
```

将文件上传到 S3 桶

以下示例显示了一个文件上传示例，其中选择使用了记录上传进度的 LoggingTransferListener。

要使用 S3 Transfer Manager 将文件上传到 Amazon S3，请将 `UploadFileRequest` 对象传递给 `S3TransferManager` 的 uploadFile 方法。

从 uploadFile 方法返回的 FileUpload 对象表示上传进程。请求完成后，CompletedFileUpload 对象将包含有关上传的信息。

```
    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!
        */
    }
```

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，其中选择使用了记录下载进度的 LoggingTransferListener。

要使用 S3 Transfer Manager 从 S3 桶下载对象，请构建 DownloadFileRequest 对象并将其传递给 downloadFile 方法。

`S3TransferManager` 的 `downloadFile` 方法返回的 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 void trackDownloadFile(S3TransferManager transferManager, String bucketName,
                                  String key, String downloadedFileWithPath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
                .getObjectRequest(b -> b.bucket(bucketName).key(key))
                .addTransferListener(LoggingTransferListener.create())  // Add
 listener.
                .destination(Paths.get(downloadedFileWithPath))
                .build();

        FileDownload downloadFile = transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult = downloadFile.completionFuture().join();
```

```
        /*
            The SDK provides a LoggingTransferListener implementation of the
TransferListener interface.
            You can also implement the interface to provide your own logic.

            Configure log4J2 with settings such as the following.
                <Configuration status="WARN">
                    <Appenders>
                        <Console name="AlignedConsoleAppender" target="SYSTEM_OUT">
                            <PatternLayout pattern="%m%n"/>
                        </Console>
                    </Appenders>

                    <Loggers>
                        <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
                            <AppenderRef ref="AlignedConsoleAppender"/>
                        </logger>
                    </Loggers>
                </Configuration>

            Log4J2 logs the progress. The following is example output for a 21.3 MB
file download.
                Transfer initiated...
                |=======             | 39.4%
                |===============     | 78.8%
                |====================| 100.0%
                Transfer complete!
        */
    }
```

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.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
```

```
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.NoSuchFileException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;
```

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。

要开始将对象从一个 S3 桶复制到另一个桶，请创建一个基本的 CopyObjectRequest 实例。

接下来，将基本 `CopyObjectRequest` 封装在可由 S3 Transfer Manager 使用的 CopyRequest 中。

`S3TransferManager` 的 `copy` 方法返回的 `Copy` 对象表示复制进程。复制进程完成后，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 Transfer Manager 执行跨区域复制，请在基于 AWS CRT 的 S3 客户端生成器上启用 `crossRegionAccessEnabled`，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` 实例的 uploadDirectory 方法，传入 UploadDirectoryRequest。

DirectoryUpload 对象表示上传进程，该进程会在请求完成时生成 CompletedDirectoryUpload。`CompleteDirectoryUpload` 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 桶中的对象下载到本地目录，请首先调用 Transfer Manager 的 downloadDirectory 方法，传入 DownloadDirectoryRequest。

DirectoryDownload 对象表示下载进程，该进程在请求完成时生成 CompletedDirectoryDownload。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 中包含本页上所有示例的完整代码。

处理 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。

您可以使用适用于 Java 的 SDK 设置存储桶，将事件发送到四个可能的目的地：

- Amazon Simple Notification Service 主题
- Amazon Simple Queue Service 队列
- AWS Lambda 函数
- Amazon EventBridge

将存储桶设置为将事件发送到 EventBridge 时，您可以配置 EventBridge 规则以将同一事件扇出到多个目的地。将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，您将了解如何使用适用于 Java 的 SDK 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。

最后一部分说明了如何使用 S3 事件通知 API 来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，queueConfigurations 方法让您可以在需要时设置多个队列目的地。此外，在 notificationConfiguration 方法中，您可以设置其他目的地，例如一个或多个 Amazon SNS 主题或一个或多个 Lambda 函数。以下代码段展示的示例中包含两个队列和三种类型的目的地。

```
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 EventBridge 的存储桶

以下示例将存储桶配置为向 EventBridge 发送通知。

```
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());
```

将存储桶配置为向 EventBridge 发送事件时，您只需指明 EventBridge 目的地，而无需指明事件类型和 EventBridge 将分派到的最终目的地。您可以使用 Java SDK 的 EventBridge 客户端配置最终目标和事件类型。

以下代码显示如何配置 EventBridge 以将已创建对象事件扇出到主题和队列。

```
   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 代码中使用 EventBridge，请在 Maven pom.xml 文件中添加 eventbridge 构件的依赖项。

```
<dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>eventbridge</artifactId>
</dependency>
```

代码示例 GitHub 存储库包含先将 S3 事件通知发送到 EventBridge，然后发送到主题和队列的完整示例。

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 API 以面向对象的方式处理这些事件。您可以使用 S3 事件通知 API 来处理直接分派到目标的事件通知（如第一个示例所示），但不能处理通过 EventBridge 路由的通知。存储桶发送到 EventBridge 的 S3 事件通知包含不同的结构，S3 事件通知 API 当前无法处理该结构。

添加依赖项

S3 事件通知 API 随适用于 Java 的 SDK 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 `pom.xml` 中，如以下代码段所示。

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>bom</artifactId>
            <version>2.X.X[1]</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
<dependencies>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>s3-event-notifications</artifactId>
    </dependency>
</dependencies>
```

[1] 最新版本。

使用 **`S3EventNotification`** 类

使用 JSON 字符串创建 **`S3EventNotification`** 实例

要将 JSON 字符串转换为 `S3EventNotification` 对象，请使用 `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，`fromJson` 方法将 JSON 字符串转换为 `S3EventNotification` 对象。JSON 字符串中缺少字段会导致相应 Java 对象字段设置为 `null` 值，而 JSON 中的任何多余字段则会被忽略。

事件通知记录的其他 API 可在 `S3EventNotificationRecord` 的 API 参考中找到。

将 `S3EventNotification` 实例转换为 JSON 字符串

使用 `toJson`（或 `toJsonPretty`）方法将 `S3EventNotification` 对象转换为 JSON 字符串，如下例所示。

```
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、ReplicationEventData、IntelligentTieringEventData 和 LifecycleEventData 的字段如果为 null，则会从 JSON 中排除。其他 null 字段将序列化为 null。

下面显示了 S3 对象标记事件的 toJsonPretty 方法示例输出。

```
{
  "Records" : [ {
    "eventVersion" : "2.3",
    "eventSource" : "aws:s3",
    "awsRegion" : "us-east-1",
    "eventTime" : "2024-07-19T20:09:18.551Z",
    "eventName" : "ObjectTagging:Put",
    "userIdentity" : {
      "principalId" : "AWS:XXXXXXXXXXX"
    },
    "requestParameters" : {
      "sourceIPAddress" : "XXX.XX.XX.XX"
    },
    "responseElements" : {
      "x-amz-request-id" : "XXXXXXXXXXXX",
      "x-amz-id-2" : "XXXXXXXXXXXXX"
    },
    "s3" : {
      "s3SchemaVersion" : "1.0",
      "configurationId" : "XXXXXXXXXXXX",
      "bucket" : {
        "name" : "amzn-s3-demo-bucket",
        "ownerIdentity" : {
          "principalId" : "XXXXXXXXXX"
        },
        "arn" : "arn:aws:s3:::XXXXXXXXX"
      },
      "object" : {
```

```
        "key" : "akey",
        "size" : null,
        "eTag" : "XXXXXXXXXX",
        "versionId" : null,
        "sequencer" : null
      }
    }
  } ]
}
```

GitHub 中提供了一个完整的示例，其中显示了如何使用 API 来处理 Amazon SQS 队列收到的通知。

使用 Java 库（AWS SDK for Java 2.x 和 **aws-lambda-java-events**）在 Lambda 中处理 S3 事件

您可以使用 3.x.x 版本的 `aws-lambda-java-events` 库，而不是使用适用于 Java 的 SDK 2.x 在 Lambda 函数中处理 Amazon S3 事件通知。AWS 单独维护 `aws-lambda-java-events` 库，并且它有自己的依赖项要求。`aws-lambda-java-events` 库仅适用于 Lambda 函数中的 S3 事件，而适用于 Java 的 SDK 2.x 可处理 Lambda 函数、Amazon SNS 和 Amazon SQS 中的 S3 事件。

两种方法都使用类似的 API 以面向对象的方式对 JSON 事件通知有效载荷建模。下表显示了使用这两种方法的显著差异。

	AWS SDK for Java	aws-lambda-java-events 库
软件包命名	`software.amazon.awssdk.eventnotifications.s3.model.S3EventNotification`	`com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification`

`RequestHandler` 参数	编写您的 Lambda 函数的 `RequestHandler` 实现以接收 JSON 字符串：
	`import com.amazonaws.services.lambda.runtime.Context;`
	`import com.amazonaws.services.lamb`
		编写您的 Lambda 函数的 `RequestHandler` 实现以接收 `S3Event` 对象：
		`import com.amazonaws.services.lambda.runtime.Context;`
		`import com.amazonaws.services.lamb`

	AWS SDK for Java	aws-lambda-java-events 库

```
da.runtime.Request
Handler;
import software.
amazon.awssdk.even
tnotifications.s3.
model.S3EventNotif
ication;

public class Handler
 implements RequestHa
ndler<String, String>
 {

    @Override
        public String
 handleRequest(String
 jsonS3Event, Context
 context) {
            S3EventNo
tification s3Event =
 S3EventNotification


   .fromJson(jsonS3Ev
ent);
            // Work with
 the s3Event object.

            ...
    }
}
```

```
da.runtime.Request
Handler;
import com.amazo
naws.services.lamb
da.runtime.events.
S3Event;

public class Handler
 implements RequestHa
ndler<S3Event, String>
 {

    @Override
        public String
 handleRequest(S3Ev
ent s3event, Context
 context) {
            // Work with
 the s3Event object.

            ...
    }
}
```

	AWS SDK for Java	aws-lambda-java-events 库
Maven 依赖项	<dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> 2.X.X</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifactId>s3- event-notifications</ artifactId> </dependency> <!-- Add other SDK dependencies that you need. --> </dependencies>	<dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> 2.X.X</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <!-- The following two dependencies are for the aws-lambda- java-events library. -- > <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-lambda-java- core</artifactId> <version> 1.2.3</version> </dependency> <dependency> <groupId> com.amazonaws</gro upId>

	AWS SDK for Java	aws-lambda-java-events 库

```
        <artifact
Id>aws-lambda-java-
events</artifactId>
        <version>
3.15.0</version>
    </dependency>
    <!-- Add other SDK
 dependencies that you
 need. -->
</dependencies>
```

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 Amazon SNS 的 `listTopics()` 方法将请求对象发送到 `SnsClient`。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 Amazon SNS 的 `publish()` 方法将请求对象发送到 `SnsClient`。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，并会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 `unsubscribe()` 的 `SnsClient` 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 Amazon SNS 的 `deleteTopic()` 方法将请求对象发送到 `SnsClient`。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。

使用 Amazon Simple Queue Service

本部分提供使用AWS SDK for Java 2.x 对 Amazon Simple Queue Service 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

- 通过AWS SDK for Java 2.x 为 Amazon SQS 使用自动请求批处理
- 处理 Amazon Simple Queue Service 消息队列
- 发送、接收和删除 Amazon Simple Queue Service 消息

通过AWS SDK for Java 2.x 为 Amazon SQS 使用自动请求批处理

Amazon SQS 的自动请求批处理 API 是一个高级库，提供了一种高效的方法来批处理和缓冲 SQS 操作的请求。通过使用批处理 API，您可以减少 SQS 的请求数量，从而提高吞吐量并更大限度地降低成本。

由于批处理 API 方法与 `SqsAsyncClient` 方法（`sendMessage`、`changeMessageVisibility`、`deleteMessage`、`receiveMessage`）相匹配，因此您可以将批处理 API 作为直接替代方案使用，只需做极少的代码修改。

本主题概述了如何配置和使用 Amazon SQS 的自动请求批处理 API。

检查先决条件

您需要使用适用于 Java 的 SDK 2.x 的版本 2.28.0 或更高版本才能访问批处理 API。您的 Maven `pom.xml` 应至少包含以下元素。

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

```
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>bom</artifactId>
            <version>2.28.23[1]</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
<dependencies>
    <dependency>
        <groupId>software.amazon.awssdk</groupId>
        <artifactId>sqs</artifactId>
    </dependency>
</dependencies>
```

[1] 最新版本

创建批量管理器

自动请求批处理 API 由 SqsAsyncBatchManager 接口实现。您可以通过几种方式创建管理器的实例。

使用 **SqsAsyncClient** 的默认配置

创建批处理管理器的最简单方法是在现有 SqsAsyncClient 实例上调用 batchManager 工厂方法。以下代码段演示了这种简单方法。

```
SqsAsyncClient asyncClient = SqsAsyncClient.create();
SqsAsyncBatchManager sqsAsyncBatchManager = asyncClient.batchManager();
```

当您使用这种方法时，SqsAsyncBatchManager 实例会使用 the section called “配置设置” 部分表格中显示的默认值。此外，SqsAsyncBatchManager 实例会使用其创建来源的 SqsAsyncClient 实例的 ExecutorService。

使用 **SqsAsyncBatchManager.Builder** 的自定义配置

对于更高级的使用案例，您可以使用 SqsAsyncBatchManager.Builder 自定义批处理管理器。通过使用这种方法来创建 SqsAsyncBatchManager 实例，您可以微调批处理行为。以下代码段举例说明了如何使用生成器来自定义批处理行为。

```
SqsAsyncBatchManager batchManager = SqsAsyncBatchManager.builder()
    .client(SqsAsyncClient.create())
```

```
    .scheduledExecutor(Executors.newScheduledThreadPool(5))
    .overrideConfiguration(b -> b
        .receiveMessageMinWaitDuration(Duration.ofSeconds(10))
        .receiveMessageVisibilityTimeout(Duration.ofSeconds(1))
        .receiveMessageAttributeNames(Collections.singletonList("*"))

 .receiveMessageSystemAttributeNames(Collections.singletonList(MessageSystemAttributeName.ALL))
    .build();
```

使用这种方法时，您可以调整 the section called “配置设置” 部分表格中显示的 BatchOverrideConfiguration 对象的设置。您也可以使用这种方法为批处理管理器提供自定义 ScheduledExecutorService。

发送消息

要使用批处理管理器发送消息，请使用 SqsAsyncBatchManager#sendMessage 方法。SDK 会缓冲请求，并在达到 maxBatchSize 或 sendRequestFrequency 值时批量发送请求。

以下示例展示了一个 sendMessage 请求后立即紧跟着另一个请求。在这种情况下，SDK 在单个批次中发送这两条消息。

```
// Sending the first message
CompletableFuture<SendMessageResponse> futureOne =
    sqsAsyncBatchManager.sendMessage(r -> r.messageBody("One").queueUrl("queue"));

// Sending the second message
CompletableFuture<SendMessageResponse> futureTwo =
    sqsAsyncBatchManager.sendMessage(r -> r.messageBody("Two").queueUrl("queue"));

// Waiting for both futures to complete and retrieving the responses
SendMessageResponse messageOne = futureOne.join();
SendMessageResponse messageTwo = futureTwo.join();
```

更改消息的可见性超时

您可以使用 SqsAsyncBatchManager#changeMessageVisibility 方法批量更改消息的可见性超时。SDK 会缓冲请求，并在达到 maxBatchSize 或 sendRequestFrequency 值时批量发送请求。

以下示例显示了如何调用 changeMessageVisibility 方法。

```
CompletableFuture<ChangeMessageVisibilityResponse> futureOne =
```

```
    sqsAsyncBatchManager.changeMessageVisibility(r ->
        r.receiptHandle("receiptHandle")
         .queueUrl("queue"));
ChangeMessageVisibilityResponse response = futureOne.join();
```

删除消息

您可以使用 SqsAsyncBatchManager#deleteMessage 方法批量删除消息。SDK 会缓冲请求，并在达到 maxBatchSize 或 sendRequestFrequency 值时批量发送请求。

以下示例说明了如何调用 deleteMessage 方法。

```
CompletableFuture<DeleteMessageResponse> futureOne =
    sqsAsyncBatchManager.deleteMessage(r ->
        r.receiptHandle("receiptHandle")
         .queueUrl("queue"));
DeleteMessageResponse response = futureOne.join();
```

接收消息

使用默认设置

当您在应用程序中轮询 SqsAsyncBatchManager#receiveMessage 方法时，批处理管理器会从其内部缓冲区获取消息，SDK 会在后台自动更新缓冲区。

以下示例显示了如何调用 receiveMessage 方法。

```
CompletableFuture<ReceiveMessageResponse> responseFuture =
    sqsAsyncBatchManager.receiveMessage(r -> r.queueUrl("queueUrl"));
```

使用自定义设置

如果您想进一步自定义请求（例如设置自定义等待时间和指定要检索的消息数量），则可以按以下示例所示自定义请求。

```
CompletableFuture<ReceiveMessageResponse> response =
    sqsAsyncBatchManager.receiveMessage(r ->
        r.queueUrl("queueUrl")
         .waitTimeSeconds(5)
         .visibilityTimeout(20));
```

> **Note**
>
> 如果您使用包括以下任何参数的 ReceiveMessageRequest 来调用 receiveMessage，则 SDK 会绕过批处理管理器并发送常规异步 receiveMessage 请求：
>
> - messageAttributeNames
> - messageSystemAttributeNames
> - messageSystemAttributeNamesWithStrings
> - overrideConfiguration

覆盖 SqsAsyncBatchManager 的配置设置

创建 SqsAsyncBatchManager 实例时，可以调整以下设置。BatchOverrideConfiguration.Builder 提供了以下设置列表。

设置	描述	默认值
maxBatchSize	每个 SendMessageBatchRequest 、ChangeMessageVisibilityBatchRequest 或 DeleteMessageBatchRequest 每批次的最大请求数。最大值为 10。	10
sendRequestFrequency	发送批次之前的时间，除非提前达到 maxBatchSize 。较高的值可能会减少请求，但会增加延迟。	200 毫秒
receiveMessageVisibilityTimeout	消息的可见性超时。如果未设置，则使用队列的默认值。	队列的默认值
receiveMessageMinWaitDuration	receiveMessage 请求的最短等待时间。避免设置为 0，以防浪费 CPU。	50 毫秒

设置	描述	默认值
receiveMessageSystemAttributeNames	receiveMessage 调用请求的系统属性名称列表。	无
receiveMessageAttributeNames	receiveMessage 调用请求的属性名称列表。	无

处理 Amazon Simple Queue Service 消息队列

消息队列是用于在中可靠地发送消息的逻辑容器 Amazon Simple Queue Service。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创建、列出、删除队列和获取 Amazon Simple Queue Service 队列的 URL AWS SDK for Java。

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();
```

使用静态 create() 方法创建 SqsClient 时，SDK 使用默认区域提供程序链配置区域，使用默认凭证提供程序链配置凭证。

创建队列

使用 SqsClient’s createQueue 方法，并提供一个描述队列参数的 CreateQueueRequest 对象，如以下代码段所示。

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’slistQueues方法。

当您使用不带任何参数的 listQueues 方法形式时，服务会返回所有队列（最多 1000 个）。

您可以向 ListQueuesRequest 对象提供一个队列名称前缀，以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's getQueueUrl 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

提供队列中指向 DeleteQueueRequest 对象的 URL。然后调用 SqsClient's deleteQueue 方法来删除队列，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();
```

使用静态 `create()` 方法创建 `SqsClient` 时，SDK 使用默认区域提供程序链配置区域，使用默认凭证提供程序链配置凭证。

发送消息

通过调用 SqsClient 客户端的 sendMessage 方法，将单个消息添加到 Amazon Simple Queue Service 队列。提供包含队列 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。此方法采用了 SendMessageBatchRequest，后者包含队列 URL 和要发送的消息的列表。（每条消息都是一个 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);
```

请参阅 GitHub 上的完整示例。

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。此方法采用了 ReceiveMessageRequest，后者包含队列 URL。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)
                .maxNumberOfMessages(5)
                .build();
            List<Message> messages =
sqsClient.receiveMessage(receiveMessageRequest).messages();
            return messages;
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
```

请参阅 GitHub 上的完整示例。

收到后删除消息

在收到消息并处理其内容后，可通过将消息的接收句柄和队列 URL 发送到 `SqsClient's` `deleteMessage` 方法，以便从队列中删除消息。

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

```
        try {
            for (Message message : messages) {
                DeleteMessageRequest deleteMessageRequest =
 DeleteMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .receiptHandle(message.receiptHandle())
                        .build();
                sqsClient.deleteMessage(deleteMessageRequest);
            }
```

请参阅 GitHub 上的完整示例。

更多信息

- 《Amazon Simple Queue Service Developer Guide》中的 How Amazon Simple Queue Service Queues Work
- 《Amazon Simple Queue Service API Reference》中的 SendMessage
- 《Amazon Simple Queue Service API Reference》中的 SendMessageBatch
- 《Amazon Simple Queue Service API Reference》中的 ReceiveMessage
- 《Amazon Simple Queue Service API Reference》中的 DeleteMessage

与... 一起工作 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。该示例使用 Amazon Transcribe 流式转录发送音频流并实时接收转录的文本流。

要详细了解此功能，请参阅 Amazon Transcribe 开发者指南中的流媒体转录。

要开始使用，请参阅 Amazon Transcribe 开发人员指南中的入门 Amazon Transcribe。

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;
    }
}
```

请参阅上的完整示例 GitHub。

创建发布者

此代码实现了一个发布者，用于发布来自音频流的 Amazon Transcribe 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();
        }
    }
}
```

请参阅上的完整示例 GitHub。

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的响应 Amazon Transcribe。

然后，使用 TranscribeStreamingAsyncClient's `startStreamTranscription` 方法开始双向流式传输。

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.StartStreamTranscriptionResponseHand
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);
        }
   }
```

请参阅上的完整示例 GitHub。

更多信息

- Amazon Transcribe 开发者指南中的@@ 工作原理。
- 《Amazon Transcribe Developer Guide》中的 Getting Started With Streaming Audio。

适用于 Java 的 SDK 2.x 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。

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

场景是向您展示如何通过在一个服务中调用多个函数或与其他 AWS 服务服务结合来完成特定任务的代码示例。

某些服务包含其他示例类别，这些类别说明如何利用特定于服务的库或函数。

Services

- 使用适用于 Java 的 SDK 2.x 的 ACM 示例
- 使用 SDK for Java 2.x 的 API Gateway 示例
- 使用适用于 Java 的 SDK 2.x 的 Application Auto Scaling 示例
- 使用 SDK for Java 2.x 的应用程序恢复控制器示例
- 使用 SDK for Java 2.x 的 Aurora 示例
- 使用 SDK for Java 2.x 的 Auto Scaling 示例
- AWS Batch 使用适用于 Java 的 SDK 2.x 的示例
- 使用 SDK for Java 2.x 的 Amazon Bedrock 示例
- 使用 SDK for Java 2.x 的 Amazon Bedrock 运行时系统示例
- CloudFront 使用适用于 Java 的 SDK 2.x 的示例
- CloudWatch 使用适用于 Java 的 SDK 2.x 的示例
- CloudWatch 使用适用于 Java 的 SDK 2.x 的事件示例
- CloudWatch 使用适用于 Java 的 SDK 2.x 记录示例
- 使用 SDK for Java 2.x 的 Amazon Cognito Identity 示例
- 使用 SDK for Java 2.x 的 Amazon Cognito 身份提供者示例
- 使用 SDK for Java 2.x 的 Amazon Comprehend 示例
- AWS Control Tower 使用适用于 Java 的 SDK 2.x 的示例
- 使用适用于 Java 的 SDK 2.x 的 Firehose 示例
- 使用适用于 Java 的 SDK 2.x 的 Amazon DocumentDB 示例

- 使用 SDK for Java 2.x 的 DynamoDB 示例
- 使用适用于 Java 的 SDK 2.x 的亚马逊 EC2 示例
- 使用适用于 Java 的 SDK 2.x 的 Amazon ECR 示例
- 使用 SDK for Java 2.x 的 Amazon ECS 示例
- ELB-使用适用于 Java 的 SDK 的第 2 版示例 2.x
- MediaStore 使用适用于 Java 的 SDK 2.x 的示例
- AWS Entity Resolution 数据匹配服务 使用适用于 Java 的 SDK 2.x 的示例
- OpenSearch 使用适用于 Java 的 SDK 2.x 的服务示例
- EventBridge 使用适用于 Java 的 SDK 2.x 的示例
- EventBridge 使用适用于 Java 的 SDK 2.x 的调度器示例
- 使用 SDK for Java 2.x 的 Forecast 示例
- 使用适用于 Java 的 SDK 2.x 的 Amazon Glacier 示例
- AWS Glue 使用适用于 Java 的 SDK 2.x 的示例
- HealthImaging 使用适用于 Java 的 SDK 2.x 的示例
- 使用 SDK for Java 2.x 的 IAM 示例
- 使用适用于 Java 的 SDK 2.x 的 Amazon Inspector 示例
- AWS IoT 使用适用于 Java 的 SDK 2.x 的示例
- AWS IoT data 使用适用于 Java 的 SDK 2.x 的示例
- AWS IoT FleetWise 使用适用于 Java 的 SDK 2.x 的示例
- AWS IoT SiteWise 使用适用于 Java 的 SDK 2.x 的示例
- 使用 SDK for Java 2.x 的 Amazon Keyspaces 示例
- 使用 SDK for Java 2.x 的 Kinesis 示例
- AWS KMS 使用适用于 Java 的 SDK 2.x 的示例
- 使用 SDK for Java 2.x 的 SDK 的 Lambda 示例
- 使用适用于 Java 的 SDK 2.x 的 Amazon Lex 示例
- 使用适用于 Java 的 SDK 2.x 的 Amazon Location 示例
- 使用适用于 Java 的 SDK 2.x 的 Location Service Places 示例
- AWS Marketplace 使用适用于 Java 的 SDK 2.x 编目 API 示例
- AWS Marketplace 使用适用于 Java 的 SDK 2.x 的协议 API 示例
- MediaConvert 使用适用于 Java 的 SDK 2.x 的示例

- 使用 SDK for Java 2.x 的 Migration Hub 示例
- 使用适用于 Java 的 SDK 2.x 的 Amazon MSK 示例
- 使用适用于 Java 的 SDK 2.x 的 Neptune 示例
- 使用适用于 Java 的 SDK 2.x 的合作伙伴中心示例
- 使用 SDK for Java 2.x 的 Amazon Personalize 示例
- 使用 SDK for Java 2.x 的 Amazon Personalize Events 示例
- 使用 SDK for Java 2.x 的 Amazon Personalize Runtime 示例
- 使用 SDK for Java 2.x 的 Amazon Pinpoint 示例
- 使用 SDK for Java 2.x 的 Amazon Pinpoint 短信和语音 API 示例
- 使用 SDK for Java 2.x 的 Amazon Polly 示例
- 使用 SDK for Java 2.x 的 Amazon RDS 示例
- 使用适用于 Java 的 SDK 2.x 的 Amazon RDS 数据服务示例
- 使用 SDK for Java 2.x 的 Amazon Redshift 示例
- 使用 SDK for Java 2.x 的 Amazon Rekognition 示例
- 使用 SDK for Java 2.x 的 Route 53 域注册示例
- 使用 SDK for Java 2.x 的 Amazon S3 示例
- 使用适用于 Java 的 SDK 2.x 的 Amazon S3 Control 示例
- 使用适用于 Java 的 SDK 2.x 的 S3 目录存储桶示例
- SageMaker 使用适用于 Java 的 SDK 2.x 的人工智能示例
- 使用 SDK for Java 2.x 的 Secrets Manager 示例
- 使用 SDK for Java 2.x 的 Amazon SES 示例
- 使用 SDK for Java 2.x 的 Amazon SES API v2 示例
- 使用 SDK for Java 2.x 的 Amazon SNS 示例
- 使用 SDK for Java 2.x 的 Amazon SQS 示例
- 使用 SDK for Java 2.x 的 Step Functions 示例
- AWS STS 使用适用于 Java 的 SDK 2.x 的示例
- 支持 使用适用于 Java 的 SDK 2.x 的示例
- 使用 SDK for Java 2.x 的 Systems Manager 示例
- 使用 SDK for Java 2.x 的 Amazon Textract 示例
- 使用 SDK for Java 2.x 的 Amazon Transcribe 示例

- 使用适用于 Java 的 SDK 2.x 的 Amazon Transcribe Streaming 示例
- 使用适用于 Java 的 SDK 2.x 的 Amazon Translate 示例

使用适用于 Java 的 SDK 2.x 的 ACM 示例

以下代码示例向您展示了如何使用 AWS SDK for Java 2.x 与 ACM 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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从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

`AddTagsToCertificate`

以下代码示例演示了如何使用 `AddTagsToCertificate`。

适用于 Java 的 SDK 2.x

> **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。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
 */
public class AddTagsToCertificate {

    public static void main(String[] args) {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        addTags(certArn);
    }

    /**
     * Adds tags to a certificate in AWS Certificate Manager (ACM).
     *
     * @param certArn the Amazon Resource Name (ARN) of the certificate to add tags
 to
     */
    public static void addTags(String certArn) {
        AcmClient acmClient = AcmClient.create();
        List<Tag> expectedTags =
 List.of(Tag.builder().key("key").value("value").build());
        AddTagsToCertificateRequest addTagsToCertificateRequest =
 AddTagsToCertificateRequest.builder()
            .certificateArn(certArn)
            .tags(expectedTags)
            .build();

        try {
            acmClient.addTagsToCertificate(addTagsToCertificateRequest);
            System.out.println("Successfully added tags to a certificate");
        } catch (AcmException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考AddTagsToCertificate中的。

`DeleteCertificate`

以下代码示例演示了如何使用 `DeleteCertificate`。

适用于 Java 的 SDK 2.x

> **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。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteCert {

    public static void main(String[] args) {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        deleteCertificate(certArn);
    }
```

```
    /**
     * Deletes an SSL/TLS certificate from the AWS Certificate Manager (ACM).
     *
     * @param certArn the Amazon Resource Name (ARN) of the certificate to be
 deleted
     */
    public static void deleteCertificate( String certArn) {
        AcmClient acmClient = AcmClient.create();
        DeleteCertificateRequest request = DeleteCertificateRequest.builder()
            .certificateArn(certArn)
            .build();

        try {
            acmClient.deleteCertificate(request);
            System.out.println("The certificate was deleted");

        } catch (AcmException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteCertificate中的。

`DescribeCertificate`

以下代码示例演示了如何使用 `DescribeCertificate`。

适用于 Java 的 SDK 2.x

> **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。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DescribeCert {

    public static void main(String[] args) {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        describeCertificate(certArn);
    }

    /**
     * Describes the details of an SSL/TLS certificate.
     *
     * @param certArn the Amazon Resource Name (ARN) of the certificate to describe
     * @throws AcmException if an error occurs while describing the certificate
     */
    public static void describeCertificate(String certArn) {
        AcmClient acmClient = AcmClient.create();
        DescribeCertificateRequest req = DescribeCertificateRequest.builder()
            .certificateArn(certArn)
            .build();

        try {
            DescribeCertificateResponse response =
 acmClient.describeCertificate(req);

            // Print the certificate details.
            System.out.println("Certificate ARN: " +
 response.certificate().certificateArn());
```

```
            System.out.println("Domain Name: " +
 response.certificate().domainName());
            System.out.println("Issued By: " + response.certificate().issuer());
            System.out.println("Issued On: " + response.certificate().issuedAt());
            System.out.println("Status: " + response.certificate().status());
        } catch (AcmException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeCertificate中的。

`ExportCertificate`

以下代码示例演示了如何使用 `ExportCertificate`。

适用于 Java 的 SDK 2.x

> **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。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ExportCertificate {

    public static void main(String[] args) throws Exception {
        final String usage = """

            Usage:    <certArn>
```

```
            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        exportCert(certArn);
    }

    /**
     * Exports an SSL/TLS certificate and its associated private key and certificate
chain from AWS Certificate Manager (ACM).
     *
     * @param certArn The Amazon Resource Name (ARN) of the certificate that you
want to export.
     * @throws IOException If an I/O error occurs while reading the private key
passphrase file or exporting the certificate.
     */
    public static void exportCert(String certArn) throws IOException {
        AcmClient acmClient = AcmClient.create();

        // Initialize a file descriptor for the passphrase file.
        RandomAccessFile filePassphrase = null;
        ByteBuffer bufPassphrase = null;

        // Create a file stream for reading the private key passphrase.
        try {
            filePassphrase = new RandomAccessFile("C:\\AWS\\password.txt", "r");
        } catch (IllegalArgumentException | SecurityException |
FileNotFoundException ex) {
            throw ex;
        }

        // Create a channel to map the file.
        FileChannel channelPassphrase = filePassphrase.getChannel();

        // Map the file to the buffer.
        try {
            bufPassphrase = channelPassphrase.map(FileChannel.MapMode.READ_ONLY, 0,
channelPassphrase.size());
            channelPassphrase.close();
```

```
            filePassphrase.close();
        } catch (IOException ex) {
            throw ex;
        }

        // Create a request object.
        ExportCertificateRequest req = ExportCertificateRequest.builder()
            .certificateArn(certArn)
            .passphrase(SdkBytes.fromByteBuffer(bufPassphrase))
            .build();

        // Export the certificate.
        ExportCertificateResponse result = null;
        try {
            result = acmClient.exportCertificate(req);
        } catch (InvalidArnException | InvalidTagException |
 ResourceNotFoundException ex) {
            throw ex;
        }

        // Clear the buffer.
        bufPassphrase.clear();

        // Display the certificate and certificate chain.
        String certificate = result.certificate();
        System.out.println(certificate);

        String certificateChain = result.certificateChain();
        System.out.println(certificateChain);

        // This example retrieves but does not display the private key.
        String privateKey = result.privateKey();
        System.out.println("The example is complete");
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ExportCertificate](#)中的。

`ImportCertificate`

以下代码示例演示了如何使用 `ImportCertificate`。

适用于 Java 的 SDK 2.x

> **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。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ImportCert {

    public static void main(String[] args) {
        final String usage = """
            Usage: <bucketName> <certificateKey> <privateKeyKey>

            Where:
                bucketName - The name of the S3 bucket containing the certificate
 and private key.
                certificateKey - The object key for the SSL/TLS certificate file in
 S3.
                privateKeyKey - The object key for the private key file in S3.
            """;

        if (args.length != 3) {
            System.out.println(usage);
            return;
        }

        String bucketName = args[0];
        String certificateKey = args[1];
        String privateKeyKey = args[2];

        String certificateArn = importCertificate(bucketName, certificateKey,
 privateKeyKey);
        System.out.println("Certificate imported with ARN: " + certificateArn);
```

```
    }

    /**
     * Imports an SSL/TLS certificate and private key from S3 into AWS Certificate
Manager (ACM).
     *
     * @param bucketName     The name of the S3 bucket.
     * @param certificateKey The key for the SSL/TLS certificate file in S3.
     * @param privateKeyKey  The key for the private key file in S3.
     * @return The ARN of the imported certificate.
     */
    public static String importCertificate(String bucketName, String certificateKey,
String privateKeyKey) {
        AcmClient acmClient = AcmClient.create();
        S3Client s3Client = S3Client.create();

        try {
            byte[] certificateBytes = downloadFileFromS3(s3Client, bucketName,
certificateKey);
            byte[] privateKeyBytes = downloadFileFromS3(s3Client, bucketName,
privateKeyKey);

            ImportCertificateRequest request = ImportCertificateRequest.builder()

.certificate(SdkBytes.fromByteBuffer(ByteBuffer.wrap(certificateBytes)))

.privateKey(SdkBytes.fromByteBuffer(ByteBuffer.wrap(privateKeyBytes)))
                    .build();

            ImportCertificateResponse response =
acmClient.importCertificate(request);
            return response.certificateArn();

        } catch (IOException e) {
            System.err.println("Error downloading certificate or private key from
S3: " + e.getMessage());
        } catch (S3Exception e) {
            System.err.println("S3 error: " + e.awsErrorDetails().errorMessage());
        }
        return "";
    }

    /**
     * Downloads a file from Amazon S3 and returns its contents as a byte array.
```

```
     *
     * @param s3Client   The S3 client.
     * @param bucketName The name of the S3 bucket.
     * @param objectKey  The key of the object in S3.
     * @return The file contents as a byte array.
     * @throws IOException If an I/O error occurs.
     */
    private static byte[] downloadFileFromS3(S3Client s3Client, String bucketName,
 String objectKey) throws IOException {
        GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

        try (ResponseInputStream<GetObjectResponse> s3Object =
 s3Client.getObject(getObjectRequest);
             ByteArrayOutputStream byteArrayOutputStream = new
 ByteArrayOutputStream()) {
            IoUtils.copy(s3Object, byteArrayOutputStream);
            return byteArrayOutputStream.toByteArray();
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ImportCertificate](中的。

`ListCertificates`

以下代码示例演示了如何使用 `ListCertificates`。

适用于 Java 的 SDK 2.x

> **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。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListCerts {
    public static void main(String[] args) {
        listCertificates();
    }

    /**
     * Lists all the certificates managed by AWS Certificate Manager (ACM) that have
 a status of "ISSUED".
     */
    public static void listCertificates() {
        AcmClient acmClient = AcmClient.create();
        try {
            ListCertificatesRequest listRequest = ListCertificatesRequest.builder()
                .certificateStatuses(CertificateStatus.ISSUED)
                .maxItems(100)
                .build();
            ListCertificatesIterable listResponse =
 acmClient.listCertificatesPaginator(listRequest);

            // Print the certificate details using streams
            listResponse.certificateSummaryList().stream()
                .forEach(certificate -> {
                    System.out.println("Certificate ARN: " +
 certificate.certificateArn());
                    System.out.println("Certificate Domain Name: " +
 certificate.domainName());
                    System.out.println("Certificate Status: " +
 certificate.statusAsString());
                    System.out.println("---");
                });

        } catch (AcmException e) {
            System.err.println(e.getMessage());
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ListCertificates](#)中的。

ListTagsForCertificate

以下代码示例演示了如何使用 ListTagsForCertificate。

适用于 Java 的 SDK 2.x

> **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。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListCertTags {

    public static void main(String[] args) {

        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        listCertTags(certArn);
    }

    /**
     * Lists the tags associated with an AWS Certificate Manager (ACM) certificate.
```

```
     *
     * @param certArn the Amazon Resource Name (ARN) of the ACM certificate
     */
    public static void listCertTags(String certArn) {
        AcmClient acmClient = AcmClient.create();

        ListTagsForCertificateRequest request =
 ListTagsForCertificateRequest.builder()
            .certificateArn(certArn)
            .build();

        ListTagsForCertificateResponse response =
 acmClient.listTagsForCertificate(request);
        List<Tag> tagList = response.tags();
        tagList.forEach(tag -> {
            System.out.println("Key: " + tag.key());
            System.out.println("Value: " + tag.value());
        });
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ListTagsForCertificate](#)中的。

`RemoveTagsFromCertificate`

以下代码示例演示了如何使用 `RemoveTagsFromCertificate`。

适用于 Java 的 SDK 2.x

> **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。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
```

```
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class RemoveTagsFromCert {

    public static void main(String[] args) {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String certArn = args[0];
        removeTags(certArn);
    }

    /**
     * Removes tags from an AWS Certificate Manager (ACM) certificate.
     *
     * @param certArn the Amazon Resource Name (ARN) of the certificate from which
 to remove tags
     */
    public static void removeTags(String certArn) {
        AcmClient acmClient = AcmClient.create();
        List<Tag> expectedTags =
List.of(Tag.builder().key("key").value("value").build());
        RemoveTagsFromCertificateRequest req =
RemoveTagsFromCertificateRequest.builder()
            .certificateArn(certArn)
            .tags(expectedTags)
            .build();

        try {
            acmClient.removeTagsFromCertificate(req);
            System.out.println("Successfully removed tags from the certificate");
        } catch (AcmException e) {
            System.err.println(e.getMessage());
```

```
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[RemoveTagsFromCertificate](#)中的。

`RenewCertificate`

以下代码示例演示了如何使用 `RenewCertificate`。

适用于 Java 的 SDK 2.x

> **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。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class RenewCert {
    public static void main(String[] args) {
        final String usage = """

            Usage:    <certArn>

            Where:
                certArn - the ARN of the certificate.
            """;
        if (args.length != 1) {
            System.out.println(usage);
            return;
```

```
        }

        String certArn = args[0];
        renewCertificate(certArn);
    }

    /**
     * Renews an existing SSL/TLS certificate in AWS Certificate Manager (ACM).
     *
     * @param certArn The Amazon Resource Name (ARN) of the certificate to be
 renewed.
     * @throws AcmException If there is an error renewing the certificate.
     */
    public static void renewCertificate(String certArn) {
        AcmClient acmClient = AcmClient.create();

        RenewCertificateRequest certificateRequest =
 RenewCertificateRequest.builder()
            .certificateArn(certArn)
            .build();

        try {
            acmClient.renewCertificate(certificateRequest);
            System.out.println("The certificate was renewed");
        } catch(AcmException e){
            System.out.println(e.getMessage());
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[RenewCertificate](#)中的。

RequestCertificate

以下代码示例演示了如何使用 RequestCertificate。

适用于 Java 的 SDK 2.x

> **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。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RequestCert {

    public static void main(String[] args) {
        requestCertificate();
    }

    /**
     * Requests a certificate from the AWS Certificate Manager (ACM) service.
     */
    public static void requestCertificate() {
        AcmClient acmClient = AcmClient.create();
        ArrayList<String> san = new ArrayList<>();
        san.add("www.example.com");

        RequestCertificateRequest req = RequestCertificateRequest.builder()
            .domainName("example.com")
            .idempotencyToken("1Aq25pTy")
            .subjectAlternativeNames(san)
            .build();

        try {
            RequestCertificateResponse response = acmClient.requestCertificate(req);
            System.out.println("Cert ARN IS " + response.certificateArn());
        } catch (AcmException e) {
            System.err.println(e.getMessage());
```

```
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考RequestCertificate中的。

使用 SDK for Java 2.x 的 API Gateway 示例

以下代码示例向您展示了如何使用 with API Gateway 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的上下文查看操作。

场景是向您展示如何通过在一个服务中调用多个函数或与其他 AWS 服务服务结合来完成特定任务的代码示例。

AWS 社区贡献就是由多个团队创建和维护的示例 AWS。要提供反馈，请使用链接存储库中提供的机制。

每个示例都包含一个指向完整源代码的链接，您可以从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
- AWS 社区捐款

操作

`CreateDeployment`

以下代码示例演示了如何使用 `CreateDeployment`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateDeployment中的。

`CreateRestApi`

以下代码示例演示了如何使用 `CreateRestApi`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateRestApi中的。

`DeleteDeployment`

以下代码示例演示了如何使用 `DeleteDeployment`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteDeployment中的。

`DeleteRestApi`

以下代码示例演示了如何使用 `DeleteRestApi`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteRestApi中的。

场景

创建无服务器应用程序来管理照片

以下代码示例演示如何创建无服务器应用程序，让用户能够使用标签管理照片。

适用于 Java 的 SDK 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 Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3

- Amazon SNS

使用 API Gateway 调用 Lambda 函数

以下代码示例展示了如何创建由 Amazon API Gateway 调用的 AWS Lambda 函数。

适用于 Java 的 SDK 2.x

演示如何使用 Lambda Java 运行时 API 创建 AWS Lambda 函数。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通过 Amazon API Gateway 调用的 Lambda 函数，该函数扫描 Amazon DynamoDB 表获取工作周年纪念日，并使用 Amazon Simple Notification Service (Amazon SNS)向员工发送文本消息，祝贺他们的周年纪念日。

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 Gateway
- DynamoDB
- Lambda
- Amazon SNS

AWS 社区捐款

构建和测试无服务器应用程序

以下代码示例演示如何使用 API Gateway 与 Lambda 和 DynamoDB 构建和测试无服务器应用程序

适用于 Java 的 SDK 2.x

演示如何使用 Java SDK 构建和测试包含 API Gateway 以及 Lambda 和 DynamoDB 的无服务器应用程序。

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

- API Gateway
- DynamoDB
- Lambda

使用适用于 Java 的 SDK 2.x 的 Application 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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从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`

以下代码示例演示了如何使用 `DeleteScalingPolicy`。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ApplicationAutoScalingExceptio
import
 software.amazon.awssdk.services.applicationautoscaling.model.DeleteScalingPolicyRequest;
import
 software.amazon.awssdk.services.applicationautoscaling.model.DeregisterScalableTargetReques
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequest
```

```
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRespons
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequest
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRespons
import
 software.amazon.awssdk.services.applicationautoscaling.model.ScalableDimension;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ServiceNamespace;

/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DisableDynamoDBAutoscaling {
    public static void main(String[] args) {
        final String usage = """

            Usage:
               <tableId> <policyName>\s

            Where:
               tableId - The table Id value (for example, table/Music).\s
               policyName - The name of the policy (for example, $Music5-scaling-
policy).

            """;
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        ApplicationAutoScalingClient appAutoScalingClient =
 ApplicationAutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        ServiceNamespace ns = ServiceNamespace.DYNAMODB;
```

```
        ScalableDimension tableWCUs =
ScalableDimension.DYNAMODB_TABLE_WRITE_CAPACITY_UNITS;
        String tableId = args[0];
        String policyName = args[1];

        deletePolicy(appAutoScalingClient, policyName, tableWCUs, ns, tableId);
        verifyScalingPolicies(appAutoScalingClient, tableId, ns, tableWCUs);
        deregisterScalableTarget(appAutoScalingClient, tableId, ns, tableWCUs);
        verifyTarget(appAutoScalingClient, tableId, ns, tableWCUs);
    }

    public static void deletePolicy(ApplicationAutoScalingClient
appAutoScalingClient, String policyName, ScalableDimension tableWCUs,
ServiceNamespace ns, String tableId) {
        try {
            DeleteScalingPolicyRequest delSPRequest =
DeleteScalingPolicyRequest.builder()
                .policyName(policyName)
                .scalableDimension(tableWCUs)
                .serviceNamespace(ns)
                .resourceId(tableId)
                .build();

            appAutoScalingClient.deleteScalingPolicy(delSPRequest);
            System.out.println(policyName +" was deleted successfully.");

        } catch (ApplicationAutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Verify that the scaling policy was deleted
    public static void verifyScalingPolicies(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
        DescribeScalingPoliciesRequest dscRequest =
DescribeScalingPoliciesRequest.builder()
            .scalableDimension(tableWCUs)
            .serviceNamespace(ns)
            .resourceId(tableId)
            .build();

        DescribeScalingPoliciesResponse response =
appAutoScalingClient.describeScalingPolicies(dscRequest);
```

```
        System.out.println("DescribeScalableTargets result: ");
        System.out.println(response);
    }

    public static void deregisterScalableTarget(ApplicationAutoScalingClient
 appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
 tableWCUs) {
        try {
            DeregisterScalableTargetRequest targetRequest =
 DeregisterScalableTargetRequest.builder()
                .scalableDimension(tableWCUs)
                .serviceNamespace(ns)
                .resourceId(tableId)
                .build();

            appAutoScalingClient.deregisterScalableTarget(targetRequest);
            System.out.println("The scalable target was deregistered.");

        } catch (ApplicationAutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    public static void verifyTarget(ApplicationAutoScalingClient
 appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
 tableWCUs) {
        DescribeScalableTargetsRequest dscRequest =
 DescribeScalableTargetsRequest.builder()
            .scalableDimension(tableWCUs)
            .serviceNamespace(ns)
            .resourceIds(tableId)
            .build();

        DescribeScalableTargetsResponse response =
 appAutoScalingClient.describeScalableTargets(dscRequest);
        System.out.println("DescribeScalableTargets result: ");
        System.out.println(response);
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteScalingPolicy中的。

RegisterScalableTarget

以下代码示例演示了如何使用 RegisterScalableTarget。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ApplicationAutoScalingExceptio
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRequest
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalableTargetsRespons
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRequest
import
 software.amazon.awssdk.services.applicationautoscaling.model.DescribeScalingPoliciesRespons
import software.amazon.awssdk.services.applicationautoscaling.model.PolicyType;
import
 software.amazon.awssdk.services.applicationautoscaling.model.PredefinedMetricSpecification;
import
 software.amazon.awssdk.services.applicationautoscaling.model.PutScalingPolicyRequest;
import
 software.amazon.awssdk.services.applicationautoscaling.model.RegisterScalableTargetRequest;
import software.amazon.awssdk.services.applicationautoscaling.model.ScalingPolicy;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ServiceNamespace;
import
 software.amazon.awssdk.services.applicationautoscaling.model.ScalableDimension;
import software.amazon.awssdk.services.applicationautoscaling.model.MetricType;
import
 software.amazon.awssdk.services.applicationautoscaling.model.TargetTrackingScalingPolicyCon
import java.util.List;

/**
```

```
 * Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class EnableDynamoDBAutoscaling {
    public static void main(String[] args) {
        final String usage = """

            Usage:
               <tableId> <roleARN> <policyName>\s

            Where:
               tableId - The table Id value (for example, table/Music).
               roleARN - The ARN of the role that has ApplicationAutoScaling
 permissions.
               policyName - The name of the policy to create.

            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        System.out.println("This example registers an Amazon DynamoDB table, which
 is the resource to scale.");
        String tableId = args[0];
        String roleARN = args[1];
        String policyName = args[2];
        ServiceNamespace ns = ServiceNamespace.DYNAMODB;
        ScalableDimension tableWCUs =
 ScalableDimension.DYNAMODB_TABLE_WRITE_CAPACITY_UNITS;
        ApplicationAutoScalingClient appAutoScalingClient =
 ApplicationAutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();

        registerScalableTarget(appAutoScalingClient, tableId, roleARN, ns,
 tableWCUs);
        verifyTarget(appAutoScalingClient, tableId, ns, tableWCUs);
```

```
        configureScalingPolicy(appAutoScalingClient, tableId, ns, tableWCUs,
policyName);
    }

    public static void registerScalableTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, String roleARN, ServiceNamespace ns,
ScalableDimension tableWCUs) {
        try {
            RegisterScalableTargetRequest targetRequest =
RegisterScalableTargetRequest.builder()
                .serviceNamespace(ns)
                .scalableDimension(tableWCUs)
                .resourceId(tableId)
                .roleARN(roleARN)
                .minCapacity(5)
                .maxCapacity(10)
                .build();

            appAutoScalingClient.registerScalableTarget(targetRequest);
            System.out.println("You have registered " + tableId);

        } catch (ApplicationAutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Verify that the target was created.
    public static void verifyTarget(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs) {
        DescribeScalableTargetsRequest dscRequest =
DescribeScalableTargetsRequest.builder()
            .scalableDimension(tableWCUs)
            .serviceNamespace(ns)
            .resourceIds(tableId)
            .build();

        DescribeScalableTargetsResponse response =
appAutoScalingClient.describeScalableTargets(dscRequest);
        System.out.println("DescribeScalableTargets result: ");
        System.out.println(response);
    }

    // Configure a scaling policy.
```

```
   public static void configureScalingPolicy(ApplicationAutoScalingClient
appAutoScalingClient, String tableId, ServiceNamespace ns, ScalableDimension
tableWCUs, String policyName) {
       // Check if the policy exists before creating a new one.
       DescribeScalingPoliciesResponse describeScalingPoliciesResponse =
appAutoScalingClient.describeScalingPolicies(DescribeScalingPoliciesRequest.builder()
           .serviceNamespace(ns)
           .resourceId(tableId)
           .scalableDimension(tableWCUs)
           .build());

       if (!describeScalingPoliciesResponse.scalingPolicies().isEmpty()) {
           // If policies exist, consider updating an existing policy instead of
creating a new one.
           System.out.println("Policy already exists. Consider updating it
instead.");
           List<ScalingPolicy> polList =
describeScalingPoliciesResponse.scalingPolicies();
           for (ScalingPolicy pol : polList) {
               System.out.println("Policy name:" +pol.policyName());
           }
       } else {
           // If no policies exist, proceed with creating a new policy.
           PredefinedMetricSpecification specification =
PredefinedMetricSpecification.builder()

.predefinedMetricType(MetricType.DYNAMO_DB_WRITE_CAPACITY_UTILIZATION)
               .build();

           TargetTrackingScalingPolicyConfiguration policyConfiguration =
TargetTrackingScalingPolicyConfiguration.builder()
               .predefinedMetricSpecification(specification)
               .targetValue(50.0)
               .scaleInCooldown(60)
               .scaleOutCooldown(60)
               .build();

           PutScalingPolicyRequest putScalingPolicyRequest =
PutScalingPolicyRequest.builder()
               .targetTrackingScalingPolicyConfiguration(policyConfiguration)
               .serviceNamespace(ns)
               .scalableDimension(tableWCUs)
               .resourceId(tableId)
               .policyName(policyName)
```

```
                .policyType(PolicyType.TARGET_TRACKING_SCALING)
                .build();

            try {
                appAutoScalingClient.putScalingPolicy(putScalingPolicyRequest);
                System.out.println("You have successfully created a scaling policy
 for an Application Auto Scaling scalable target");
            } catch (ApplicationAutoScalingException e) {
                System.err.println("Error: " + e.awsErrorDetails().errorMessage());
            }
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考RegisterScalableTarget中的。

使用 SDK for 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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从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`

以下代码示例演示了如何使用 `GetRoutingControlState`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetRoutingControlState中的。

`UpdateRoutingControlState`

以下代码示例演示了如何使用 `UpdateRoutingControlState`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考UpdateRoutingControlState中的。

使用 SDK for 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。

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 基本功能
- 操作
- 场景

开始使用

开始使用 Aurora

以下代码示例显示如何开始使用 Aurora。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DBClusters中的描述。

基本功能

了解基本功能

以下代码示例展示了如何：

- 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。

适用于 Java 的 SDK 2.x

> **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。

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including 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《AWS SDK for Java 2.x API Reference》中的以下主题。
  - 创建DBCluster
  - 创建DBClusterParameterGroup
  - 创建DBCluster快照
  - 创建DBInstance
  - 删除DBCluster
  - 删除DBClusterParameterGroup
  - 删除DBInstance
  - 描述DBClusterParameterGroups
  - 描述DBCluster参数
  - 描述DBCluster快照
  - 描述DBClusters
  - 描述DBEngine版本
  - 描述DBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

操作

`CreateDBCluster`

以下代码示例演示了如何使用 `CreateDBCluster`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅DBCluster在 AWS SDK for Java 2.x API 参考中创建。

`CreateDBClusterParameterGroup`

以下代码示例演示了如何使用 `CreateDBClusterParameterGroup`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅DBClusterParameterGroup在 AWS SDK for Java 2.x API 参考中创建。

`CreateDBClusterSnapshot`

以下代码示例演示了如何使用 `CreateDBClusterSnapshot`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 “创建DBCluster快照”。

`CreateDBInstance`

以下代码示例演示了如何使用 `CreateDBInstance`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅DBInstance在 AWS SDK for Java 2.x API 参考中创建。

`DeleteDBCluster`

以下代码示例演示了如何使用 `DeleteDBCluster`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅DBCluster《AWS SDK for Java 2.x API 参考》中的 “删除”。

DeleteDBClusterParameterGroup

以下代码示例演示了如何使用 DeleteDBClusterParameterGroup。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅DBClusterParameterGroup《AWS SDK for Java 2.x API 参考》中的 “删除”。

`DeleteDBInstance`

以下代码示例演示了如何使用 `DeleteDBInstance`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅DBInstance《AWS SDK for Java 2.x API 参考》中的 “删除”。

`DescribeDBClusterParameterGroups`

以下代码示例演示了如何使用 `DescribeDBClusterParameterGroups`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DBClusterParameterGroups中的描述。

`DescribeDBClusterParameters`

以下代码示例演示了如何使用 `DescribeDBClusterParameters`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 “描述DBCluster参数”。

DescribeDBClusterSnapshots

以下代码示例演示了如何使用 DescribeDBClusterSnapshots。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 “描述DBCluster快照”。

`DescribeDBClusters`

以下代码示例演示了如何使用 `DescribeDBClusters`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DBClusters中的描述。

`DescribeDBEngineVersions`

以下代码示例演示了如何使用 `DescribeDBEngineVersions`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 “描述DBEngine版本”。

`DescribeDBInstances`

以下代码示例演示了如何使用 `DescribeDBInstances`。

适用于 Java 的 SDK 2.x

> **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。

```
   // 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DBInstances中的描述。

DescribeOrderableDBInstanceOptions

以下代码示例演示了如何使用 DescribeOrderableDBInstanceOptions。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考中的DescribeOrderableDBInstance选项。

ModifyDBClusterParameterGroup

以下代码示例演示了如何使用 ModifyDBClusterParameterGroup。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅《AWS SDK for Java 2.x API 参考DBClusterParameterGroup》中的 “修改”。

场景

创建 Aurora Serverless 工作项跟踪器

以下代码示例演示如何创建 Web 应用程序，来跟踪 Amazon Aurora Serverless 数据库中的工作项，以及使用 Amazon Simple Email Service（Amazon SES）发送报告。

适用于 Java 的 SDK 2.x

展示如何创建 Web 应用程序来跟踪与报告存储与 Amazon RDS 数据库的工作项。

有关如何设置查询 Amazon Aurora Serverless 数据的 Spring REST API 以及如何让 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。

有关如何设置和运行使用 JDBC API 的示例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

- Aurora
- Amazon RDS
- Amazon RDS 数据服务
- Amazon SES

使用 SDK for 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。

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用

- 基本功能
- 操作
- 场景

开始使用

开始使用 Auto Scaling

以下代码示例显示了如何开始使用 Auto Scaling。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeAutoScalingGroups中的。

基本功能

了解基本功能

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。

适用于 Java 的 SDK 2.x

> **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。

```
/**
 * Before running this SDK for Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, create a launch template. For more information, see the
 * following topic:
 *
 * https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-launch-
templates.html#create-launch-template
 *
 * This code example performs the following operations:
 * 1. Creates an Auto Scaling group using an AutoScalingWaiter.
 * 2. Gets a specific Auto Scaling group and returns an instance Id value.
 * 3. Describes Auto Scaling with the Id value.
 * 4. Enables metrics collection.
 * 5. Update an Auto Scaling group.
 * 6. Describes Account details.
 * 7. Describe account details"
 * 8. Updates an Auto Scaling group to use an additional instance.
 * 9. Gets the specific Auto Scaling group and gets the number of instances.
 * 10. List the scaling activities that have occurred for the group.
 * 11. Terminates an instance in the Auto Scaling group.
 * 12. Stops the metrics collection.
 * 13. Deletes the Auto Scaling group.
 */

public class AutoScalingScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final String ROLES_STACK = "MyCdkAutoScaleStack";
    public static void main(String[] args) throws InterruptedException {
        final String usage = """

                Usage:
                    <groupName>

                Where:
                    groupName - The name of the Auto Scaling group.
                """;
```

```
        String groupName = "MyAutoScalingGroup2";
        AutoScalingClient autoScalingClient = AutoScalingClient.builder()
                .region(Region.US_WEST_2)
                .build();

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

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon EC2 Auto Scaling example
scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("First, we will create a launch template using a
CloudFormation script");
        CloudFormationHelper.deployCloudFormationStack(ROLES_STACK);
        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputsAsync(ROLES_STACK).join();
        String launchTemplateName = stackOutputs.get("LaunchTemplateNameOutput");
        String vpcZoneId = getVPC(ec2);
        updateTemlate(ec2, launchTemplateName );
        System.out.println("The VPC zone id created by the CloudFormation stack
is"+vpcZoneId);

        System.out.println("1. Create an Auto Scaling group named " + groupName);
        createAutoScalingGroup(autoScalingClient, ec2, groupName,
launchTemplateName, vpcZoneId);

        System.out.println(
                "Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned");
        Thread.sleep(60000);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Get Auto Scale group Id value");
        String instanceId = getSpecificAutoScalingGroups(autoScalingClient,
groupName);
        if (instanceId.compareTo("") == 0) {
            System.out.println("Error - no instance Id value");
            System.exit(1);
        } else {
```

```
            System.out.println("The instance Id value is " + instanceId);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Describe Auto Scaling with the Id value " +
instanceId);
        describeAutoScalingInstance(autoScalingClient, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Enable metrics collection " + instanceId);
        enableMetricsCollection(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Update an Auto Scaling group to update max size to
3");
        updateAutoScalingGroup(autoScalingClient, groupName, launchTemplateName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Describe Auto Scaling groups");
        describeAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Describe account details");
        describeAccountLimits(autoScalingClient);
        System.out.println(
                "Wait 1 min for the resources, including the instance. Otherwise, an
empty instance Id is returned");
        Thread.sleep(60000);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Set desired capacity to 2");
        setDesiredCapacity(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get the two instance Id values and state");
        getSpecificAutoScalingGroups(autoScalingClient, groupName);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("10. List the scaling activities that have occurred for
the group");
        describeScalingActivities(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Terminate an instance in the Auto Scaling group");
        terminateInstanceInAutoScalingGroup(autoScalingClient, instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Stop the metrics collection");
        disableMetricsCollection(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Delete the Auto Scaling group and cloud formation
resources");
        CloudFormationHelper.destroyCloudFormationStack(ROLES_STACK);
        deleteAutoScalingGroup(autoScalingClient, groupName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Scenario has successfully completed.");
        System.out.println(DASHES);

        autoScalingClient.close();
    }

    public static String getVPC(Ec2Client ec2) {
        try {
            DescribeVpcsRequest request = DescribeVpcsRequest.builder()
                    .filters(f -> f.name("isDefault").values("true"))
                    .build();

            DescribeVpcsResponse response = ec2.describeVpcs(request);

            if (!response.vpcs().isEmpty()) {
                Vpc defaultVpc = response.vpcs().get(0);
                System.out.println("Default VPC ID: " + defaultVpc.vpcId());
                return defaultVpc.vpcId();
            } else {
```

```
                System.out.println("No default VPC found.");
                return null; // Return null if no default VPC is found
            }

        } catch (Ec2Exception e) {
            System.err.println("EC2 error: " + e.awsErrorDetails().errorMessage());
            return null; // Return null in case of an error
        }
    }


    public static void updateTemlate(Ec2Client ec2, String launchTemplateName ) {
        // Step 1: Create new launch template version
        String newAmiId = "ami-0025f0db847eb6254";
        RequestLaunchTemplateData launchTemplateData =
RequestLaunchTemplateData.builder()
                .imageId(newAmiId)
                .build();

        CreateLaunchTemplateVersionRequest createVersionRequest =
CreateLaunchTemplateVersionRequest.builder()
                .launchTemplateName(launchTemplateName)
                .versionDescription("Updated with valid AMI")
                .sourceVersion("1")
                .launchTemplateData(launchTemplateData)
                .build();

        CreateLaunchTemplateVersionResponse createResponse =
ec2.createLaunchTemplateVersion(createVersionRequest);
        int newVersionNumber =
createResponse.launchTemplateVersion().versionNumber().intValue();

        // Step 2: Modify default version
        ModifyLaunchTemplateRequest modifyRequest =
ModifyLaunchTemplateRequest.builder()
                .launchTemplateName(launchTemplateName)
                .defaultVersion(String.valueOf(newVersionNumber))
                .build();

        ec2.modifyLaunchTemplate(modifyRequest);
        System.out.println("Updated launch template to version " + newVersionNumber
+ " with AMI " + newAmiId);
    }
```

```
   public static void describeScalingActivities(AutoScalingClient
autoScalingClient, String groupName) {
       try {
           DescribeScalingActivitiesRequest scalingActivitiesRequest =
DescribeScalingActivitiesRequest.builder()
                   .autoScalingGroupName(groupName)
                   .maxRecords(10)
                   .build();

           DescribeScalingActivitiesResponse response = autoScalingClient
                   .describeScalingActivities(scalingActivitiesRequest);
           List<Activity> activities = response.activities();
           for (Activity activity : activities) {
               System.out.println("The activity Id is " + activity.activityId());
               System.out.println("The activity details are " +
activity.details());
           }

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   public static void setDesiredCapacity(AutoScalingClient autoScalingClient,
String groupName) {
       try {
           SetDesiredCapacityRequest capacityRequest =
SetDesiredCapacityRequest.builder()
                   .autoScalingGroupName(groupName)
                   .desiredCapacity(2)
                   .build();

           autoScalingClient.setDesiredCapacity(capacityRequest);
           System.out.println("You have set the DesiredCapacity to 2");

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   public static void createAutoScalingGroup(AutoScalingClient autoScalingClient,
```

```
                                            Ec2Client ec2Client,
                                            String groupName,
                                            String launchTemplateName,
                                            String vpcId) {
        try {
            // Step 1: Get one subnet ID in the given VPC
            DescribeSubnetsRequest subnetRequest = DescribeSubnetsRequest.builder()
                    .filters(Filter.builder().name("vpc-id").values(vpcId).build())
                    .build();

            DescribeSubnetsResponse subnetResponse =
ec2Client.describeSubnets(subnetRequest);

            if (subnetResponse.subnets().isEmpty()) {
                throw new RuntimeException("No subnets found in VPC: " + vpcId);
            }

            String subnetId = subnetResponse.subnets().get(0).subnetId(); // Use
first subnet
            System.out.println("Using subnet: " + subnetId);

            // Step 2: Create launch template reference
            LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                    .launchTemplateName(launchTemplateName)
                    .build();

            // Step 3: Create Auto Scaling group
            CreateAutoScalingGroupRequest request =
CreateAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .launchTemplate(templateSpecification)
                    .minSize(1)
                    .maxSize(1)
                    .vpcZoneIdentifier(subnetId)  // Correct: subnet ID, not VPC ID
                    .build();

            autoScalingClient.createAutoScalingGroup(request);

            // Step 4: Wait until group is created
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
```

```
                    .build();

            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
                    waiter.waitUntilGroupExists(groupsRequest);

            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Auto Scaling Group created");

        } catch (Ec2Exception | AutoScalingException e) {
            System.err.println("Error: " + e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void describeAutoScalingInstance(AutoScalingClient
autoScalingClient, String id) {
        try {
            DescribeAutoScalingInstancesRequest describeAutoScalingInstancesRequest
= DescribeAutoScalingInstancesRequest
                    .builder()
                    .instanceIds(id)
                    .build();

            DescribeAutoScalingInstancesResponse response = autoScalingClient

.describeAutoScalingInstances(describeAutoScalingInstancesRequest);
            List<AutoScalingInstanceDetails> instances =
response.autoScalingInstances();
            for (AutoScalingInstanceDetails instance : instances) {
                System.out.println("The instance lifecycle state is: " +
instance.lifecycleState());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void describeAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
```

```
                    .autoScalingGroupNames(groupName)
                    .maxRecords(10)
                    .build();

            DescribeAutoScalingGroupsResponse response =
autoScalingClient.describeAutoScalingGroups(groupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
                System.out.println("*** The service to use for the health checks: "
+ group.healthCheckType());
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getSpecificAutoScalingGroups(AutoScalingClient
autoScalingClient, String groupName) {
        try {
            String instanceId = "";
            DescribeAutoScalingGroupsRequest scalingGroupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();

            DescribeAutoScalingGroupsResponse response = autoScalingClient
                    .describeAutoScalingGroups(scalingGroupsRequest);
            List<AutoScalingGroup> groups = response.autoScalingGroups();
            for (AutoScalingGroup group : groups) {
                System.out.println("The group name is " +
group.autoScalingGroupName());
                System.out.println("The group ARN is " +
group.autoScalingGroupARN());
                List<Instance> instances = group.instances();

                for (Instance instance : instances) {
                    instanceId = instance.instanceId();
                    System.out.println("The instance id is " + instanceId);
                    System.out.println("The lifecycle state is " +
instance.lifecycleState());
                }
            }
```

```
            return instanceId;
        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void enableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            EnableMetricsCollectionRequest collectionRequest =
EnableMetricsCollectionRequest.builder()
                    .autoScalingGroupName(groupName)
                    .metrics("GroupMaxSize")
                    .granularity("1Minute")
                    .build();

            autoScalingClient.enableMetricsCollection(collectionRequest);
            System.out.println("The enable metrics collection operation was
successful");

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void disableMetricsCollection(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            DisableMetricsCollectionRequest disableMetricsCollectionRequest =
DisableMetricsCollectionRequest.builder()
                    .autoScalingGroupName(groupName)
                    .metrics("GroupMaxSize")
                    .build();


autoScalingClient.disableMetricsCollection(disableMetricsCollectionRequest);
            System.out.println("The disable metrics collection operation was
successful");

        } catch (AutoScalingException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void describeAccountLimits(AutoScalingClient autoScalingClient) {
        try {
            DescribeAccountLimitsResponse response =
autoScalingClient.describeAccountLimits();
            System.out.println("The max number of auto scaling groups is " +
response.maxNumberOfAutoScalingGroups());
            System.out.println("The current number of auto scaling groups is " +
response.numberOfAutoScalingGroups());

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void updateAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName,
            String launchTemplateName) {
        try {
            AutoScalingWaiter waiter = autoScalingClient.waiter();
            LaunchTemplateSpecification templateSpecification =
LaunchTemplateSpecification.builder()
                    .launchTemplateName(launchTemplateName)
                    .build();

            UpdateAutoScalingGroupRequest groupRequest =
UpdateAutoScalingGroupRequest.builder()
                    .maxSize(3)
                    .autoScalingGroupName(groupName)
                    .launchTemplate(templateSpecification)
                    .build();

            autoScalingClient.updateAutoScalingGroup(groupRequest);
            DescribeAutoScalingGroupsRequest groupsRequest =
DescribeAutoScalingGroupsRequest.builder()
                    .autoScalingGroupNames(groupName)
                    .build();
```

```
            WaiterResponse<DescribeAutoScalingGroupsResponse> waiterResponse =
waiter
                    .waitUntilGroupInService(groupsRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("You successfully updated the auto scaling group  " +
groupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void terminateInstanceInAutoScalingGroup(AutoScalingClient
autoScalingClient, String instanceId) {
        try {
            TerminateInstanceInAutoScalingGroupRequest request =
TerminateInstanceInAutoScalingGroupRequest.builder()
                    .instanceId(instanceId)
                    .shouldDecrementDesiredCapacity(false)
                    .build();

            autoScalingClient.terminateInstanceInAutoScalingGroup(request);
            System.out.println("You have terminated instance " + instanceId);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteAutoScalingGroup(AutoScalingClient autoScalingClient,
String groupName) {
        try {
            DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                    .autoScalingGroupName(groupName)
                    .forceDelete(true)
                    .build();

            autoScalingClient.deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
            System.out.println("You successfully deleted " + groupName);

        } catch (AutoScalingException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

操作

`CreateAutoScalingGroup`

以下代码示例演示了如何使用 `CreateAutoScalingGroup`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateAutoScalingGroup中的。

`DeleteAutoScalingGroup`

以下代码示例演示了如何使用 `DeleteAutoScalingGroup`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteAutoScalingGroup中的。

DescribeAutoScalingGroups

以下代码示例演示了如何使用 DescribeAutoScalingGroups。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeAutoScalingGroups中的。

`DescribeAutoScalingInstances`

以下代码示例演示了如何使用 `DescribeAutoScalingInstances`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeAutoScalingInstances中的。

`DescribeScalingActivities`

以下代码示例演示了如何使用 `DescribeScalingActivities`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeScalingActivities中的。

DisableMetricsCollection

以下代码示例演示了如何使用 DisableMetricsCollection。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DisableMetricsCollection中的。

EnableMetricsCollection

以下代码示例演示了如何使用 EnableMetricsCollection。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考EnableMetricsCollection中的。

SetDesiredCapacity

以下代码示例演示了如何使用 SetDesiredCapacity。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SetDesiredCapacity中的。

`TerminateInstanceInAutoScalingGroup`

以下代码示例演示了如何使用 `TerminateInstanceInAutoScalingGroup`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[TerminateInstanceInAutoScalingGroup](#)中的。

`UpdateAutoScalingGroup`

以下代码示例演示了如何使用 `UpdateAutoScalingGroup`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，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 (Amazon EC2) 实例，并将实例数量保持在指定范围内。
- 使用弹性负载均衡处理和分发 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。

适用于 Java 的 SDK 2.x

> **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 wasSuccessul = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group for
your default VPC must
                            allow access from this computer. You can either add it
automatically from this
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
```

```
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            System.out.println("Your load balancer is ready. You can access it by
browsing to:");
            System.out.println("\t http://" + elbDnsName);
        } else {
            System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
            System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
            System.out.println("you can successfully make a GET request to the load
balancer.");
        }

        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
```

```
                """
                        This part of the demonstration shows how to toggle
different parts of the system
                        to create situations where the web service fails, and shows
how using a resilient
                        architecture can keep the web service running in spite of
these failures.

                        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
                        """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                        The table name is contained in a Systems Manager parameter
named self.param_helper.table.
                        To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
                        """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println(
                """
                        \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
                        healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
                        """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        Instead of failing when the recommendation service fails,
the web service can return a static response.
                        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
                        """);
        paramHelper.put(paramHelper.failureResponse, "static");

        System.out.println("""
```

```
                Now, sending a GET request to the load balancer endpoint returns a
static response.
                The service still reports as healthy because health checks are still
shallow.
                """);
        demoChoices(loadBalancer);

        System.out.println("Let's reinstate the recommendation service.");
        paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

        System.out.println("""
                Let's also substitute bad credentials for one of the instances in
the target group so that it can't
                access the DynamoDB recommendation table. We will get an instance id
value.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        AutoScaler autoScaler = new AutoScaler();

        // Create a new instance profile based on badCredsProfileName.
        templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
        String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
        System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

        String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is " + profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId
                + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

        System.out.println(
                """
                        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
                        depending on which instance is selected by the load
balancer.
                        """);

        demoChoices(loadBalancer);
```

```
        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
check tests whether
                the web service can access the DynamoDB table that it depends on for
recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
Scaling instance health.
                This kind of deep health check is not recommended for Auto Scaling
instance health, because it
                risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
                """);

        System.out.println("""
                By implementing deep health checks, the load balancer can detect
when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
                Now, checking target health indicates that the instance with bad
credentials
                is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
                instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
                """);

        demoChoices(loadBalancer);

        System.out.println(
                """
                        Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
                        instance is to terminate it and let the auto scaler start a
new instance to replace it.
                        """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
```

```
                Even while the instance is terminating and the new instance is
starting, sending a GET
                request to the web service continues to get a successful
recommendation response because
                the load balancer routes requests to the healthy instances. After
the replacement instance
                starts and reports as healthy, it is included in the load balancing
rotation.
                Note that terminating and replacing an instance typically takes
several minutes, during which time you
                can see the changing health check status until the new instance is
running and healthy.
                """);

        demoChoices(loadBalancer);
        System.out.println(
                "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        String[] actions = {
                "Send a GET request to the load balancer endpoint.",
                "Check the health of load balancer targets.",
                "Go to the next part of the demo."
        };
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
```

```
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
HttpClients.createDefault();

                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
                        HttpResponse response = httpClient.execute(httpGet);
                        int statusCode = response.getStatusLine().getStatusCode();
                        System.out.println("HTTP Status Code: " + statusCode);

                        // Display the JSON response
                        BufferedReader reader = new BufferedReader(
                                new
InputStreamReader(response.getEntity().getContent()));
                        StringBuilder jsonResponse = new StringBuilder();
                        String line;
                        while ((line = reader.readLine()) != null) {
                            jsonResponse.append(line);
                        }
                        reader.close();

                        // Print the formatted JSON response.
                        System.out.println("Full Response:\n");
                        System.out.println(jsonResponse.toString());

                        // Close the HTTP client.
                        httpClient.close();

                    }
                    case 1 -> {
                        System.out.println("\nChecking the health of load balancer
targets:\n");
                        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
                        for (TargetHealthDescription target : health) {
```

```
                            System.out.printf("\tTarget %s on port %d is %s%n",
 target.target().id(),
                                    target.target().port(),
 target.targetHealth().stateAsString());
                        }
                        System.out.println("""
                                Note that it can take a minute or two for the health
 check to update
                                after changes are made.
                                """);
                    }
                    case 2 -> {
                        System.out.println("\nOkay, let's move on.");
                        System.out.println("-".repeat(88));
                        return; // Exit the method when choice is 2
                    }
                    default -> System.out.println("You must choose a value between
 0-2. Please select again.");
                }

            } catch (java.util.InputMismatchException e) {
                System.out.println("Invalid input. Please select again.");
                scanner.nextLine(); // Clear the input buffer.
            }
        }
    }

    public static String readFileAsString(String filePath) throws IOException {
        byte[] bytes = Files.readAllBytes(Paths.get(filePath));
        return new String(bytes);
    }
}
```

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.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
                .instanceIds(instanceId)
                .documentName("AWS-RunShellScript")
                .parameters(Collections.singletonMap("commands",
                        Collections.singletonList("cd / && sudo python3 server.py
80")))
                .build();
```

```
        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress) {
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(secGroupId)
                .cidrIp(ipAddress)
                .fromPort(Integer.parseInt(port))
                .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                    .builder()
                    .instanceProfileName(profileName)
                    .build();

            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .roleName(roleName)
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
```

```
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .build();

            getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
            System.out.println("Deleted instance profile " + profileName);

            DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            // List attached role policies.
            ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                    .listAttachedRolePolicies(role -> role.roleName(roleName));
            List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
            for (AttachedPolicy attachedPolicy : attachedPolicies) {
                DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(attachedPolicy.policyArn())
                        .build();

                getIAMClient().detachRolePolicy(request);
                System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
            }

            getIAMClient().deleteRole(deleteRoleRequest);
            System.out.println("Instance profile and role deleted.");

        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
```

```
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();


getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network, you
     * must instead specify a prefix list ID. You can also temporarily open the port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     *
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                    .name("group-name")
                    .values("default")
                    .build();

            Filter filter1 = Filter.builder()
                    .name("vpc-id")
                    .values(VPC)
                    .build();

            DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
                    .filters(filter, filter1)
                    .build();

            DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
                    .describeSecurityGroups(securityGroupsRequest);
```

```
            String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
            groupInfo.setGroupName(securityGroup);

            for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
                System.out.println("Found security group: " + secGroup.groupId());

                for (IpPermission ipPermission : secGroup.ipPermissions()) {
                    if (ipPermission.fromPort() == port) {
                        System.out.println("Found inbound rule: " + ipPermission);
                        for (IpRange ipRange : ipPermission.ipRanges()) {
                            String cidrIp = ipRange.cidrIp();
                            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                                System.out.println(cidrIp + " is applicable");
                                portIsOpen = true;
                            }
                        }

                        if (!ipPermission.prefixListIds().isEmpty()) {
                            System.out.println("Prefix lList is applicable");
                            portIsOpen = true;
                        }

                        if (!portIsOpen) {
                            System.out
                                    .println("The inbound rule does not appear to be
open to either this computer's IP,"
                                            + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
                        } else {
                            break;
                        }
                    }
                }
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }

        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }
```

```
   /*
    * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
    * Scaling group.
    * The target group specifies how the load balancer forward requests to the
    * instances
    * in the group.
    */
   public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
       try {
           AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                   .autoScalingGroupName(asGroupName)
                   .targetGroupARNs(targetGroupARN)
                   .build();


getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
           System.out.println("Attached load balancer to " + asGroupName);

       } catch (AutoScalingException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   // Creates an EC2 Auto Scaling group with the specified size.
   public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

       // Get availability zones.
       software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
               .builder()
               .build();

       DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
       List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
```

```
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
                .launchTemplate(specification)
                .availabilityZones(zones)
                .maxSize(groupSize)
                .minSize(groupSize)
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }

    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();

        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
                .builder()
                .filters(defaultFilter)
                .build();
```

```
        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
                .name("vpc-id")
                .values(vpcId)
                .build();

        Filter azFilter = Filter.builder()
                .name("availability-zone")
                .values(availabilityZones)
                .build();

        Filter defaultForAZ = Filter.builder()
                .name("default-for-az")
                .values("true")
                .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
                .filters(vpcFilter, azFilter, defaultForAZ)
                .build();

        DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

        DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
                .map(instance -> instance.instanceId())
                .collect(Collectors.toList());
```

```
        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }

    // Gets data about the profile associated with an instance.
    public String getInstanceProfile(String instanceId) {
        Filter filter = Filter.builder()
                .name("instance-id")
                .values(instanceId)
                .build();

        DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
                .builder()
                .filters(filter)
                .build();

        DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
                .describeIamInstanceProfileAssociations(associationsRequest);
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
        ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to the
policy.

software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                        .builder()
                        .policyArn(policy.arn())
                        .build();
```

```
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                        .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                        || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
 DetachRolePolicyRequest.builder()
                            .policyArn(policy.arn())
                            .roleName(roleName) // Specify the name of the IAM role
                            .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
 DeletePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .build();

                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
                break;
            }
        }

        // List the roles associated with the instance profile
        ListInstanceProfilesForRoleRequest listRolesRequest =
 ListInstanceProfilesForRoleRequest.builder()
                .roleName(roleName)
                .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
 iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
 listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
 RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(InstanceProfile)
                    .roleName(roleName) // Remove the extra dot here
                    .build();
```

```
            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance profile
 " + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
 DeleteInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .build();

        getIAMClient().deleteInstanceProfile(r ->
 r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

创建一个包装 ELB 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)
                    .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《AWS SDK for Java 2.x API Reference》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

AWS Batch 使用适用于 Java 的 SDK 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 Batch。 AWS SDK for Java 2.x

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 基本功能
- 操作

开始使用

你好 AWS Batch

以下代码示例展示了如何开始使用 AWS Batch。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.batch.BatchAsyncClient;
import software.amazon.awssdk.services.batch.model.JobStatus;
import software.amazon.awssdk.services.batch.model.JobSummary;
import software.amazon.awssdk.services.batch.model.ListJobsRequest;
import software.amazon.awssdk.services.batch.paginators.ListJobsPublisher;
import java.time.Duration;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CompletableFuture;

public class HelloBatch {
    private static BatchAsyncClient batchClient;

    public static void main(String[] args) {
        List<JobSummary> jobs = listJobs("my-job-queue");
        jobs.forEach(job ->
            System.out.printf("Job ID: %s, Job Name: %s, Job Status: %s%n",
                job.jobId(), job.jobName(), job.status())
        );
    }

    public static List<JobSummary> listJobs(String jobQueue) {
        if (jobQueue == null || jobQueue.isEmpty()) {
            throw new IllegalArgumentException("Job queue cannot be null or empty");
        }

        ListJobsRequest listJobsRequest = ListJobsRequest.builder()
            .jobQueue(jobQueue)
            .jobStatus(JobStatus.SUCCEEDED)
```

```
            .build();

        List<JobSummary> jobSummaries = new ArrayList<>();
        ListJobsPublisher listJobsPaginator =
getAsyncClient().listJobsPaginator(listJobsRequest);
        CompletableFuture<Void> future = listJobsPaginator.subscribe(response -> {
            jobSummaries.addAll(response.jobSummaryList());
        });

        future.join();
        return jobSummaries;
    }

    private static BatchAsyncClient getAsyncClient() {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)  // Increase max concurrency to handle more
simultaneous connections.
            .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
            .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
            .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
timeout.
            .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the individual
call attempt timeout.
            .retryPolicy(RetryPolicy.builder()  // Add a retry policy to handle
transient errors.
                .numRetries(3)  // Number of retry attempts.
                .build())
            .build();

        if (batchClient == null) {
            batchClient = BatchAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return batchClient;
    }
```

```
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考listJobsPaginator中的。

基本功能

了解基本功能

以下代码示例展示了如何：

- 创建 AWS Batch 计算环境。
- 检查计算环境的状态。
- 设置 AWS Batch 作业队列和作业定义。
- 注册作业定义。
- 提 AWS Batch 交 Job。
- 获取适用于作业队列的作业列表。
- 检查作业的状态。
- 删除 AWS Batch 资源。

适用于 Java 的 SDK 2.x

> **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。

运行演示 AWS Batch 功能的交互式场景。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.batch.model.BatchException;
import software.amazon.awssdk.services.batch.model.ClientException;
import software.amazon.awssdk.services.batch.model.CreateComputeEnvironmentResponse;
import software.amazon.awssdk.services.batch.model.JobSummary;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsRequest;
```

```
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeSubnetsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeSubnetsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest;
import software.amazon.awssdk.services.ec2.model.Filter;
import software.amazon.awssdk.services.ec2.model.SecurityGroup;
import software.amazon.awssdk.services.ec2.model.Subnet;
import software.amazon.awssdk.services.ec2.model.Vpc;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * NOTE
 * This scenario submits a job that pulls a Docker image named echo-text from Amazon
 ECR to Amazon Fargate.
 *
 * To place this Docker image on Amazon ECR, run the following Basics scenario.
 *
 * https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/ecr
 *
 */
public class BatchScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    // Define two stacks used in this Basics Scenario.
    private static final String ROLES_STACK = "RolesStack";
    private static String defaultSubnet;
    private static String defaultSecurityGroup;

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

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

        BatchActions batchActions = new BatchActions();
        Scanner scanner = new Scanner(System.in);
        String computeEnvironmentName = "my-compute-environment";
        String jobQueueName = "my-job-queue";
        String jobDefinitionName = "my-job-definition";


        // See the NOTE in this Java code example (at start).
        String dockerImage = "dkr.ecr.us-east-1.amazonaws.com/echo-text:echo-text";

        logger.info("""
            AWS Batch is a fully managed batch processing service that dynamically
 provisions the required compute
            resources for batch computing workloads. The Java V2 `BatchAsyncClient`
 allows
            developers to automate the submission, monitoring, and management of
 batch jobs.

            This scenario provides an example of setting up a compute environment,
 job queue and job definition,
            and then submitting a job.

            This scenario submits a job that pulls a Docker image named echo-text
 from Amazon ECR to Amazon Fargate.

            To place this Docker image on Amazon ECR, run the following Basics
 scenario.

            https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/
example_code/ecr

            Let's get started...

            You have two choices:

            1 - Run the entire program.
            2 - Delete an existing Compute Environment (created from a previous
 execution of
            this program that did not complete).
            """);
```

```
        while (true) {
            String input = scanner.nextLine();
            if (input.trim().equalsIgnoreCase("1")) {
                logger.info("Continuing with the program...");
               // logger.info("");
                break;
            } else if (input.trim().equalsIgnoreCase("2")) {
                String jobQueueARN = String.valueOf(batchActions.
describeJobQueueAsync(computeEnvironmentName));
                if (!jobQueueARN.isEmpty()) {
                    batchActions.disableJobQueueAsync(jobQueueARN);
                    countdown(1);
                    batchActions.deleteJobQueueAsync(jobQueueARN);
                }

                try {

batchActions.disableComputeEnvironmentAsync(computeEnvironmentName)
                        .exceptionally(ex -> {
                            logger.info("Disable compute environment failed: " +
ex.getMessage());
                            return null;
                        })
                        .join();
                } catch (CompletionException ex) {
                    logger.info("Failed to disable compute environment: " +
ex.getMessage());
                }
                countdown(2);

batchActions.deleteComputeEnvironmentAsync(computeEnvironmentName).join();
                return;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
        System.out.println(DASHES);

        waitForInputToContinue(scanner);
        // Get an AWS Account id used to retrieve the docker image from Amazon ECR.
        // Create a single-element array to store the `accountId` value.
        String[] accId = new String[1];
        CompletableFuture<String> accountIdFuture = batchActions.getAccountId();
```

```
        accountIdFuture.thenAccept(accountId -> {
            logger.info("Account ID: " + accountId);
            accId[0] = accountId;
        }).join();

        dockerImage = accId[0]+"."+dockerImage;

        // Get a default subnet and default security associated with the default
VPC.
        getSubnetSecurityGroup();

        logger.info("Use AWS CloudFormation to create two IAM roles that are
required for this scenario.");
        CloudFormationHelper.deployCloudFormationStack(ROLES_STACK);

        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputs(ROLES_STACK);
        String batchIAMRole = stackOutputs.get("BatchRoleArn");
        String executionRoleARN = stackOutputs.get("EcsRoleArn");

        logger.info("The IAM role needed to interact with AWS Batch is
"+batchIAMRole);
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("1. Create a Batch compute environment");
        logger.info("""
            A compute environment is a resource where you can run your batch jobs.
            After creating a compute environment, you can define job queues and job
definitions to submit jobs for
            execution.

            The benefit of creating a compute environment is it allows you to easily
configure and manage the compute
            resources that will be used to run your Batch jobs. By separating the
compute environment from the job definitions,
            you can easily scale your compute resources up or down as needed,
without having to modify your job definitions.
            This makes it easier to manage your Batch workloads and ensures that
your jobs have the necessary
            compute resources to run efficiently.
            """);

        waitForInputToContinue(scanner);
```

```
        try {
            CompletableFuture<CreateComputeEnvironmentResponse> future =
batchActions.createComputeEnvironmentAsync(computeEnvironmentName, batchIAMRole,
defaultSubnet, defaultSecurityGroup);
            CreateComputeEnvironmentResponse response = future.join();
            logger.info("Compute Environment ARN: " +
response.computeEnvironmentArn());
        } catch (RuntimeException rte) {
            Throwable cause = rte.getCause();
            if (cause instanceof ClientException batchExceptionEx) {
                String myErrorCode =
batchExceptionEx.awsErrorDetails().errorMessage();
                if ("Object already exists".contains(myErrorCode)) {
                    logger.info("The compute environment '" + computeEnvironmentName
+ "' already exists. Moving on...");
                } else {
                    logger.info("Batch error occurred: {} (Code: {})",
batchExceptionEx.getMessage(), batchExceptionEx.awsErrorDetails().errorCode());
                    return;
                }
            } else {
                    logger.info("An unexpected error occurred: {}", (cause != null ?
cause.getMessage() : rte.getMessage()));
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Check the status of the "+computeEnvironmentName +" Compute
Environment.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
batchActions.checkComputeEnvironmentsStatus(computeEnvironmentName);
            String status = future.join();
            logger.info("Compute Environment Status: " + status);

        } catch (RuntimeException rte) {
            Throwable cause = rte.getCause();
            if (cause instanceof ClientException batchExceptionEx) {
                logger.info("Batch error occurred: {} (Code: {})",
batchExceptionEx.getMessage(), batchExceptionEx.awsErrorDetails().errorCode());
                return;
```

```
            } else {
                logger.info("An unexpected error occurred: " + (cause != null ?
cause.getMessage() : rte.getMessage()));
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Create a job queue");
        logger.info("""
             A job queue is an essential component that helps manage the execution
of your batch jobs.
             It acts as a buffer, where jobs are placed and then scheduled for
execution based on their
             priority and the available resources in the compute environment.
             """);
        waitForInputToContinue(scanner);

        String jobQueueArn = null;
        try {
            CompletableFuture<String> jobQueueFuture =
batchActions.createJobQueueAsync(jobQueueName, computeEnvironmentName);
            jobQueueArn = jobQueueFuture.join();
            logger.info("Job Queue ARN: " + jobQueueArn);

        } catch (RuntimeException rte) {
            Throwable cause = rte.getCause();
            if (cause instanceof BatchException batchExceptionEx) {
                String myErrorCode =
batchExceptionEx.awsErrorDetails().errorMessage();
                if ("Object already exists".contains(myErrorCode)) {
                    logger.info("The job queue '" + jobQueueName + "' already
exists. Moving on...");
                    // Retrieve the ARN of the job queue.
                    CompletableFuture<String> jobQueueArnFuture =
batchActions.getJobQueueARN(jobQueueName);
                    jobQueueArn = jobQueueArnFuture.join();
                    logger.info("Job Queue ARN: " + jobQueueArn);
                } else {
                    logger.info("Batch error occurred: {} (Code: {})",
batchExceptionEx.getMessage(), batchExceptionEx.awsErrorDetails().errorCode());
                    return;
```

```
                }
            } else {
                logger.info("An unexpected error occurred: " + (cause != null ?
cause.getMessage() : rte.getMessage()));
                return; // End the execution
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("4. Register a Job Definition.");
        logger.info("""
            Registering a job in AWS Batch using the Fargate launch type ensures
that all
            necessary parameters, such as the execution role, command to run, and so
on
            are specified and reused across multiple job submissions.

             The job definition pulls a Docker image from Amazon ECR and executes
the Docker image.
            """);

        waitForInputToContinue(scanner);
        String jobARN;
        try {
            String platform = "";
            while (true) {
                logger.info("""
                    On which platform/CPU architecture combination did you build the
Docker image?:
                    1. Windows       X86_64
                    2. Mac or Linux  ARM64
                    3. Mac or Linux  X86_64

                    Please select 1, 2, or 3.
                    """);
                String platAns = scanner.nextLine().trim();
                if (platAns.equals("1")) {
                    platform = "X86_64";
                    break; // Exit loop since a valid option is selected
                } else if (platAns.equals("2")) {
                    platform = "ARM64";
                    break; // Exit loop since a valid option is selected
                } else if (platAns.equals("3")) {
```

```
                    platform = "X86_64";
                    break; // Exit loop since a valid option is selected
                } else {
                    System.out.println("Invalid input. Please select either 1 or
2.");
                }
            }

            jobARN = batchActions.registerJobDefinitionAsync(jobDefinitionName,
executionRoleARN, dockerImage, platform)
                .exceptionally(ex -> {
                    System.err.println("Register job definition failed: " +
ex.getMessage());
                    return null;
                })
                .join();
            if (jobARN != null) {
                logger.info("Job ARN: " + jobARN);
            }
        } catch (RuntimeException rte) {
            logger.error("A Batch exception occurred while registering the job: {}",
rte.getCause() != null ? rte.getCause().getMessage() : rte.getMessage());
            return;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Submit an AWS Batch job from a job definition.");
        waitForInputToContinue(scanner);
        String jobId;
        try {
            jobId = batchActions.submitJobAsync(jobDefinitionName, jobQueueName,
jobARN)
                .exceptionally(ex -> {
                    System.err.println("Submit job failed: " + ex.getMessage());
                    return null;
                })
                .join();

            logger.info("The job id is "+jobId);
            logger.info("Let's wait 2 minutes for the job to complete");
            countdown(2);

        } catch (RuntimeException rte) {
```

```
            logger.error("A Batch exception occurred while submitting the job: {}",
rte.getCause() != null ? rte.getCause().getMessage() : rte.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        logger.info(DASHES);
        logger.info("6. Get a list of jobs applicable to the job queue.");

        waitForInputToContinue(scanner);
        try {
            List<JobSummary> jobs = batchActions.listJobsAsync(jobQueueName);
            jobs.forEach(job ->
                logger.info("Job ID: {}, Job Name: {}, Job Status: {}", job.jobId(),
job.jobName(), job.status()));

        } catch (RuntimeException rte) {
            logger.info("A Batch exception occurred while submitting the job: {}",
rte.getCause() != null ? rte.getCause().getMessage() : rte.getMessage());
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Check the status of job "+jobId);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future = batchActions.describeJobAsync(jobId);
            String jobStatus = future.join();
            logger.info("Job Status: " + jobStatus);

        } catch (RuntimeException rte) {
            logger.info("A Batch exception occurred while submitting the job: {}",
rte.getCause() != null ? rte.getCause().getMessage() : rte.getMessage());
            return;
        }

        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        logger.info("8. Delete Batch resources");
```

```
        logger.info(
            """
            When deleting an AWS Batch compute environment, it does not happen
instantaneously.
            There is typically a delay, similar to some other AWS resources.
            AWS Batch starts the deletion process.
            """);
        logger.info("Would you like to delete the AWS Batch resources such as the
compute environment? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete the AWS ECR resources.");
            logger.info("First, we will deregister the Job Definition.");
            waitForInputToContinue(scanner);
            try {
                batchActions.deregisterJobDefinitionAsync(jobARN)
                    .exceptionally(ex -> {
                        logger.info("Deregister job definition failed: " +
ex.getMessage());
                        return null;
                    })
                    .join();
                logger.info(jobARN + " was deregistered");
            } catch (RuntimeException rte) {
                logger.error("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }

            logger.info("Second, we will disable and then delete the Job Queue.");
            waitForInputToContinue(scanner);
            try {
                batchActions.disableJobQueueAsync(jobQueueArn)
                    .exceptionally(ex -> {
                        logger.info("Disable job queue failed: " + ex.getMessage());
                        return null;
                    })
                    .join();
                logger.info(jobQueueArn + " was disabled");
            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }
```

```
            batchActions.waitForJobQueueToBeDisabledAsync(jobQueueArn);
            try {
                CompletableFuture<Void> future =
batchActions.waitForJobQueueToBeDisabledAsync(jobQueueArn);
                future.join();
                logger.info("Job queue is now disabled.");
            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }

            waitForInputToContinue(scanner);
            try {
                batchActions.deleteJobQueueAsync(jobQueueArn);
                logger.info(jobQueueArn +" was deleted");
            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }
            logger.info("Let's wait 2 minutes for the job queue to be deleted");
            countdown(2);
            waitForInputToContinue(scanner);

            logger.info("Third, we will delete the Compute Environment.");
            waitForInputToContinue(scanner);
            try {
                batchActions.disableComputeEnvironmentAsync(computeEnvironmentName)
                    .exceptionally(ex -> {
                        System.err.println("Disable compute environment failed: " +
ex.getMessage());
                        return null;
                    })
                    .join();
                logger.info("Compute environment disabled") ;
            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }
```

```
batchActions.checkComputeEnvironmentsStatus(computeEnvironmentName).thenAccept(state
-> {
                logger.info("Current State: " + state);
            }).join();

            logger.info("Lets wait 1 min for the compute environment to be
deleted");
            countdown(1);

            try {

batchActions.deleteComputeEnvironmentAsync(computeEnvironmentName).join();
                logger.info(computeEnvironmentName +" was deleted.");

            } catch (RuntimeException rte) {
                logger.info("A Batch exception occurred: {}", rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage());
                return;
            }
            waitForInputToContinue(scanner);
            CloudFormationHelper.destroyCloudFormationStack(ROLES_STACK);
        }

        logger.info(DASHES);
        logger.info("This concludes the AWS Batch SDK scenario");
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
```

```
    }

    public static void countdown(int minutes) throws InterruptedException {
        int seconds = 0;
        for (int i = minutes * 60 + seconds; i >= 0; i--) {
            int displayMinutes = i / 60;
            int displaySeconds = i % 60;
            System.out.print(String.format("\r%02d:%02d", displayMinutes,
displaySeconds));
            Thread.sleep(1000); // Wait for 1 second
        }
        logger.info("Countdown complete!");
    }

    private static void getSubnetSecurityGroup() {
        try (Ec2AsyncClient ec2Client = Ec2AsyncClient.create()) {
            CompletableFuture<Vpc> defaultVpcFuture =
ec2Client.describeVpcs(DescribeVpcsRequest.builder()
                            .filters(Filter.builder()
                                    .name("is-default")
                                    .values("true")
                                    .build())
                            .build())
                    .thenApply(response -> response.vpcs().stream()
                            .findFirst()
                            .orElseThrow(() -> new RuntimeException("Default VPC not
found")));

            CompletableFuture<String> defaultSubnetFuture = defaultVpcFuture
                    .thenCompose(vpc ->
ec2Client.describeSubnets(DescribeSubnetsRequest.builder()
                                    .filters(Filter.builder()
                                                    .name("vpc-id")
                                                    .values(vpc.vpcId())
                                                    .build(),
                                            Filter.builder()
                                                    .name("default-for-az")
                                                    .values("true")
                                                    .build())
                                    .build())
                            .thenApply(DescribeSubnetsResponse::subnets)
                            .thenApply(subnets -> subnets.stream()
                                    .findFirst()
                                    .map(Subnet::subnetId)
```

```
                                    .orElseThrow(() -> new RuntimeException("No
 default subnet found"))));

            CompletableFuture<String> defaultSecurityGroupFuture = defaultVpcFuture
                    .thenCompose(vpc ->
 ec2Client.describeSecurityGroups(DescribeSecurityGroupsRequest.builder()
                                    .filters(Filter.builder()
                                                    .name("group-name")
                                                    .values("default")
                                                    .build(),
                                            Filter.builder()
                                                    .name("vpc-id")
                                                    .values(vpc.vpcId())
                                                    .build())
                                    .build())

 .thenApply(DescribeSecurityGroupsResponse::securityGroups)
                            .thenApply(securityGroups -> securityGroups.stream()
                                    .findFirst()
                                    .map(SecurityGroup::groupId)
                                    .orElseThrow(() -> new RuntimeException("No
 default security group found"))));

            defaultSubnet = defaultSubnetFuture.join();
            defaultSecurityGroup = defaultSecurityGroupFuture.join();
        }
    }
}
```

AWS Batch SDK 方法的包装器类。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.batch.BatchAsyncClient;
import software.amazon.awssdk.services.batch.BatchClient;
import software.amazon.awssdk.services.batch.model.AssignPublicIp;
import software.amazon.awssdk.services.batch.model.BatchException;
```

```
import software.amazon.awssdk.services.batch.model.CEState;
import software.amazon.awssdk.services.batch.model.CEType;
import software.amazon.awssdk.services.batch.model.CRType;
import software.amazon.awssdk.services.batch.model.ComputeEnvironmentOrder;
import software.amazon.awssdk.services.batch.model.ComputeResource;
import software.amazon.awssdk.services.batch.model.ContainerProperties;
import software.amazon.awssdk.services.batch.model.CreateComputeEnvironmentRequest;
import software.amazon.awssdk.services.batch.model.CreateComputeEnvironmentResponse;
import software.amazon.awssdk.services.batch.model.CreateJobQueueRequest;
import software.amazon.awssdk.services.batch.model.DeleteComputeEnvironmentRequest;
import software.amazon.awssdk.services.batch.model.DeleteComputeEnvironmentResponse;
import software.amazon.awssdk.services.batch.model.DeleteJobQueueRequest;
import software.amazon.awssdk.services.batch.model.DeleteJobQueueResponse;
import software.amazon.awssdk.services.batch.model.DeregisterJobDefinitionRequest;
import software.amazon.awssdk.services.batch.model.DeregisterJobDefinitionResponse;
import
 software.amazon.awssdk.services.batch.model.DescribeComputeEnvironmentsRequest;
import
 software.amazon.awssdk.services.batch.model.DescribeComputeEnvironmentsResponse;
import software.amazon.awssdk.services.batch.model.DescribeJobQueuesRequest;
import software.amazon.awssdk.services.batch.model.DescribeJobQueuesResponse;
import software.amazon.awssdk.services.batch.model.DescribeJobsRequest;
import software.amazon.awssdk.services.batch.model.DescribeJobsResponse;
import software.amazon.awssdk.services.batch.model.JQState;
import software.amazon.awssdk.services.batch.model.JobDefinitionType;
import software.amazon.awssdk.services.batch.model.JobDetail;
import software.amazon.awssdk.services.batch.model.JobQueueDetail;
import software.amazon.awssdk.services.batch.model.JobStatus;
import software.amazon.awssdk.services.batch.model.JobSummary;
import software.amazon.awssdk.services.batch.model.ListJobsRequest;
import software.amazon.awssdk.services.batch.model.RegisterJobDefinitionResponse;
import software.amazon.awssdk.services.batch.model.NetworkConfiguration;
import software.amazon.awssdk.services.batch.model.PlatformCapability;
import software.amazon.awssdk.services.batch.model.RegisterJobDefinitionRequest;
import software.amazon.awssdk.services.batch.model.ResourceRequirement;
import software.amazon.awssdk.services.batch.model.ResourceType;
import software.amazon.awssdk.services.batch.model.RuntimePlatform;
import software.amazon.awssdk.services.batch.model.SubmitJobRequest;
import software.amazon.awssdk.services.batch.model.CreateJobQueueResponse;
import java.time.Duration;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.List;
```

```
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.atomic.AtomicBoolean;
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.services.batch.model.SubmitJobResponse;
import software.amazon.awssdk.services.batch.model.UpdateComputeEnvironmentRequest;
import software.amazon.awssdk.services.batch.model.UpdateComputeEnvironmentResponse;
import software.amazon.awssdk.services.batch.model.UpdateJobQueueRequest;
import software.amazon.awssdk.services.batch.model.UpdateJobQueueResponse;
import software.amazon.awssdk.services.batch.paginators.ListJobsPublisher;
import software.amazon.awssdk.services.sts.StsAsyncClient;
import software.amazon.awssdk.services.sts.model.GetCallerIdentityResponse;

public class BatchActions {
    private static BatchAsyncClient batchClient;

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

    private static BatchAsyncClient getAsyncClient() {
        if (batchClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryPolicy(RetryPolicy.builder()
                    .numRetries(3)
                    .build())
                .build();

            batchClient = BatchAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return batchClient;
```

```
    }

    /**
     * Asynchronously creates a new compute environment in AWS Batch.
     *
     * @param computeEnvironmentName the name of the compute environment to create
     * @param batchIAMRole the IAM role to be used by the compute environment
     * @param subnet the subnet ID to be used for the compute environment
     * @param secGroup the security group ID to be used for the compute environment
     * @return a {@link CompletableFuture} representing the asynchronous operation,
which will complete with the
     *         {@link CreateComputeEnvironmentResponse} when the compute environment
has been created
     * @throws BatchException if there is an error creating the compute environment
     * @throws RuntimeException if there is an unexpected error during the operation
     */
    public CompletableFuture<CreateComputeEnvironmentResponse>
createComputeEnvironmentAsync(
        String computeEnvironmentName, String batchIAMRole, String subnet, String
secGroup) {
        CreateComputeEnvironmentRequest environmentRequest =
CreateComputeEnvironmentRequest.builder()
            .computeEnvironmentName(computeEnvironmentName)
            .type(CEType.MANAGED)
            .state(CEState.ENABLED)
            .computeResources(ComputeResource.builder()
                .type(CRType.FARGATE)
                .maxvCpus(256)
                .subnets(Collections.singletonList(subnet))
                .securityGroupIds(Collections.singletonList(secGroup))
                .build())
            .serviceRole(batchIAMRole)
            .build();

        CompletableFuture<CreateComputeEnvironmentResponse> response =
getAsyncClient().createComputeEnvironment(environmentRequest);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                throw new RuntimeException(errorMessage, ex);
            }
        });
```

```
        return response;
    }

    public CompletableFuture<DeleteComputeEnvironmentResponse>
deleteComputeEnvironmentAsync(String computeEnvironmentName) {
        DeleteComputeEnvironmentRequest deleteComputeEnvironment =
DeleteComputeEnvironmentRequest.builder()
            .computeEnvironment(computeEnvironmentName)
            .build();

        return getAsyncClient().deleteComputeEnvironment(deleteComputeEnvironment)
            .whenComplete((response, ex) -> {
                if (ex != null) {
                    Throwable cause = ex.getCause();
                    if (cause instanceof BatchException) {
                        throw new RuntimeException(cause);
                    } else {
                        throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
                    }
                }
            });
    }

    /**
     * Checks the status of the specified compute environment.
     *
     * @param computeEnvironmentName the name of the compute environment to check
     * @return a CompletableFuture containing the status of the compute environment,
or "ERROR" if an exception occurs
     */
    public CompletableFuture<String> checkComputeEnvironmentsStatus(String
computeEnvironmentName) {
        if (computeEnvironmentName == null || computeEnvironmentName.isEmpty()) {
            throw new IllegalArgumentException("Compute environment name cannot be
null or empty");
        }

        DescribeComputeEnvironmentsRequest environmentsRequest =
DescribeComputeEnvironmentsRequest.builder()
            .computeEnvironments(computeEnvironmentName)
            .build();
```

```
        CompletableFuture<DescribeComputeEnvironmentsResponse> response =
getAsyncClient().describeComputeEnvironments(environmentsRequest);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                throw new RuntimeException(errorMessage, ex);
            }
        });

        return response.thenApply(resp -> resp.computeEnvironments().stream()
            .map(env -> env.statusAsString())
            .findFirst()
            .orElse("UNKNOWN"));
    }

    /**
     * Creates a job queue asynchronously.
     *
     * @param jobQueueName the name of the job queue to create
     * @param computeEnvironmentName the name of the compute environment to
associate with the job queue
     * @return a CompletableFuture that completes with the Amazon Resource Name
(ARN) of the job queue
     */
    public CompletableFuture<String> createJobQueueAsync(String jobQueueName, String
computeEnvironmentName) {
        if (jobQueueName == null || jobQueueName.isEmpty()) {
            throw new IllegalArgumentException("Job queue name cannot be null or
empty");
        }
        if (computeEnvironmentName == null || computeEnvironmentName.isEmpty()) {
            throw new IllegalArgumentException("Compute environment name cannot be
null or empty");
        }

        CreateJobQueueRequest request = CreateJobQueueRequest.builder()
            .jobQueueName(jobQueueName)
            .priority(1)
            .computeEnvironmentOrder(ComputeEnvironmentOrder.builder()
                .computeEnvironment(computeEnvironmentName)
                .order(1)
                .build())
            .build();
```

```
        CompletableFuture<CreateJobQueueResponse> response =
getAsyncClient().createJobQueue(request);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                throw new RuntimeException(errorMessage, ex);
            }
        });

        return response.thenApply(CreateJobQueueResponse::jobQueueArn);
    }

    /**
     * Asynchronously lists the jobs in the specified job queue with the given job
status.
     *
     * @param jobQueue the name of the job queue to list jobs from
     * @return a List<JobSummary> that contains the jobs that succeeded
     */
    public List<JobSummary> listJobsAsync(String jobQueue) {
        if (jobQueue == null || jobQueue.isEmpty()) {
            throw new IllegalArgumentException("Job queue cannot be null or empty");
        }

        ListJobsRequest listJobsRequest = ListJobsRequest.builder()
            .jobQueue(jobQueue)
            .jobStatus(JobStatus.SUCCEEDED)  // Filter jobs by status.
            .build();

        List<JobSummary> jobSummaries = new ArrayList<>();
        ListJobsPublisher listJobsPaginator =
getAsyncClient().listJobsPaginator(listJobsRequest);
        CompletableFuture<Void> future = listJobsPaginator.subscribe(response -> {
            jobSummaries.addAll(response.jobSummaryList());
        });
        future.join();
        return jobSummaries;
    }

    /**
     * Registers a new job definition asynchronously in AWS Batch.
     * <p>
```

```
    * When using Fargate as the compute environment, it is crucial to set the
    * {@link NetworkConfiguration} with {@link AssignPublicIp#ENABLED} to
    * ensure proper networking configuration for the Fargate tasks. This
    * allows the tasks to communicate with external services, access the
    * internet, or communicate within a VPC.
    *
    * @param jobDefinitionName the name of the job definition to be registered
    * @param executionRoleARN the ARN (Amazon Resource Name) of the execution role
    *                         that provides permissions for the containers in the
job
    * @param cpuArch a value of either X86_64 or ARM64 required for the service
call
    * @return a CompletableFuture that completes with the ARN of the registered
    *         job definition upon successful execution, or completes exceptionally
with
    *         an error if the registration fails
    */
   public CompletableFuture<String> registerJobDefinitionAsync(String
jobDefinitionName, String executionRoleARN, String image, String cpuArch) {
       NetworkConfiguration networkConfiguration = NetworkConfiguration.builder()
               .assignPublicIp(AssignPublicIp.ENABLED)
               .build();

       ContainerProperties containerProperties = ContainerProperties.builder()
               .image(image)
               .executionRoleArn(executionRoleARN)
               .resourceRequirements(
                       Arrays.asList(
                               ResourceRequirement.builder()
                                       .type(ResourceType.VCPU)
                                       .value("1")
                                       .build(),
                               ResourceRequirement.builder()
                                       .type(ResourceType.MEMORY)
                                       .value("2048")
                                       .build()
                       )
               )
               .networkConfiguration(networkConfiguration)
              .runtimePlatform(b -> b
                       .cpuArchitecture(cpuArch)
                       .operatingSystemFamily("LINUX"))
               .build();
```

```
        RegisterJobDefinitionRequest request =
RegisterJobDefinitionRequest.builder()
                .jobDefinitionName(jobDefinitionName)
                .type(JobDefinitionType.CONTAINER)
                .containerProperties(containerProperties)
                .platformCapabilities(PlatformCapability.FARGATE)
                .build();

        CompletableFuture<String> future = new CompletableFuture<>();
        getAsyncClient().registerJobDefinition(request)
                .thenApply(RegisterJobDefinitionResponse::jobDefinitionArn)
                .whenComplete((result, ex) -> {
                    if (ex != null) {
                        future.completeExceptionally(ex);
                    } else {
                        future.complete(result);
                    }
                });

        return future;
    }

    /**
     * Deregisters a job definition asynchronously.
     *
     * @param jobDefinition the name of the job definition to be deregistered
     * @return a CompletableFuture that completes when the job definition has been
deregistered
     * or an exception has occurred
     */
    public CompletableFuture<DeregisterJobDefinitionResponse>
deregisterJobDefinitionAsync(String jobDefinition) {
        DeregisterJobDefinitionRequest jobDefinitionRequest =
DeregisterJobDefinitionRequest.builder()
            .jobDefinition(jobDefinition)
            .build();

        CompletableFuture<DeregisterJobDefinitionResponse> responseFuture =
getAsyncClient().deregisterJobDefinition(jobDefinitionRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
            }
```

```
        });

        return responseFuture;
    }

    /**
     * Disables the specified job queue asynchronously.
     *
     * @param jobQueueArn the Amazon Resource Name (ARN) of the job queue to be
disabled
     * @return a {@link CompletableFuture} that completes when the job queue update
operation is complete,
     *         or completes exceptionally if an error occurs during the operation
     */
    public CompletableFuture<Void> disableJobQueueAsync(String jobQueueArn) {
        UpdateJobQueueRequest updateRequest = UpdateJobQueueRequest.builder()
            .jobQueue(jobQueueArn)
            .state(JQState.DISABLED)
            .build();

        CompletableFuture<UpdateJobQueueResponse> responseFuture =
getAsyncClient().updateJobQueue(updateRequest);
        return responseFuture.whenComplete((updateResponse, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to update job queue: " +
ex.getMessage(), ex);
            }
        }).thenApply(updateResponse -> null);
    }

    /**
     * Deletes a Batch job queue asynchronously.
     *
     * @param jobQueueArn The Amazon Resource Name (ARN) of the job queue to delete.
     * @return A CompletableFuture that represents the asynchronous deletion of the
job queue.
     *         The future completes when the job queue has been successfully deleted
or if an error occurs.
     *         If successful, the future will be completed with a {@code Void}
value.
     *         If an error occurs, the future will be completed exceptionally with
the thrown exception.
     */
    public CompletableFuture<Void> deleteJobQueueAsync(String jobQueueArn) {
```

```
        DeleteJobQueueRequest deleteRequest = DeleteJobQueueRequest.builder()
            .jobQueue(jobQueueArn)
            .build();

        CompletableFuture<DeleteJobQueueResponse> responseFuture =
getAsyncClient().deleteJobQueue(deleteRequest);
        return responseFuture.whenComplete((deleteResponse, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete job queue: " +
ex.getMessage(), ex);
            }
        }).thenApply(deleteResponse -> null);
    }

    /**
     * Asynchronously describes the job queue associated with the specified compute
environment.
     *
     * @param computeEnvironmentName the name of the compute environment to find the
associated job queue for
     * @return a {@link CompletableFuture} that, when completed, contains the job
queue ARN associated with the specified compute environment
     * @throws RuntimeException if the job queue description fails
     */
    public CompletableFuture<String> describeJobQueueAsync(String
computeEnvironmentName) {
        DescribeJobQueuesRequest describeJobQueuesRequest =
DescribeJobQueuesRequest.builder()
            .build();

        CompletableFuture<DescribeJobQueuesResponse> responseFuture =
getAsyncClient().describeJobQueues(describeJobQueuesRequest);
        return responseFuture.whenComplete((describeJobQueuesResponse, ex) -> {
            if (describeJobQueuesResponse != null) {
                String jobQueueARN;
                for (JobQueueDetail jobQueueDetail :
describeJobQueuesResponse.jobQueues()) {
                    for (ComputeEnvironmentOrder computeEnvironmentOrder :
jobQueueDetail.computeEnvironmentOrder()) {
                        String computeEnvironment =
computeEnvironmentOrder.computeEnvironment();
                        String name = getComputeEnvironmentName(computeEnvironment);
                        if (name.equals(computeEnvironmentName)) {
                            jobQueueARN = jobQueueDetail.jobQueueArn();
```

```
                            logger.info("Job queue ARN associated with the compute
environment: " + jobQueueARN);
                        }
                    }
                }
            } else {
                throw new RuntimeException("Failed to describe job queue: " +
ex.getMessage(), ex);
            }
        }).thenApply(describeJobQueuesResponse -> {
            String jobQueueARN = "";
            for (JobQueueDetail jobQueueDetail :
describeJobQueuesResponse.jobQueues()) {
                for (ComputeEnvironmentOrder computeEnvironmentOrder :
jobQueueDetail.computeEnvironmentOrder()) {
                    String computeEnvironment =
computeEnvironmentOrder.computeEnvironment();
                    String name = getComputeEnvironmentName(computeEnvironment);
                    if (name.equals(computeEnvironmentName)) {
                        jobQueueARN = jobQueueDetail.jobQueueArn();
                    }
                }
            }
            return jobQueueARN;
        });
    }

    /**
     * Disables the specified compute environment asynchronously.
     *
     * @param computeEnvironmentName the name of the compute environment to disable
     * @return a CompletableFuture that completes when the compute environment is
disabled
     */
    public CompletableFuture<UpdateComputeEnvironmentResponse>
disableComputeEnvironmentAsync(String computeEnvironmentName) {
        UpdateComputeEnvironmentRequest updateRequest =
UpdateComputeEnvironmentRequest.builder()
            .computeEnvironment(computeEnvironmentName)
            .state(CEState.DISABLED)
            .build();

        CompletableFuture<UpdateComputeEnvironmentResponse> responseFuture =
getAsyncClient().updateComputeEnvironment(updateRequest);
```

```
        responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to disable compute environment: "
+ ex.getMessage(), ex);
            }
        });

        return responseFuture;
    }

    /**
     * Submits a job asynchronously to the AWS Batch service.
     *
     * @param jobDefinitionName the name of the job definition to use
     * @param jobQueueName the name of the job queue to submit the job to
     * @param jobARN the Amazon Resource Name (ARN) of the job definition
     * @return a CompletableFuture that, when completed, contains the job ID of the
submitted job
     */
    public CompletableFuture<String> submitJobAsync(String jobDefinitionName, String
jobQueueName, String jobARN) {
        SubmitJobRequest jobRequest = SubmitJobRequest.builder()
            .jobDefinition(jobARN)
            .jobName(jobDefinitionName)
            .jobQueue(jobQueueName)
            .build();

        CompletableFuture<SubmitJobResponse> responseFuture =
getAsyncClient().submitJob(jobRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
            }
        });

        return responseFuture.thenApply(SubmitJobResponse::jobId);
    }

    /**
     * Asynchronously retrieves the status of a specific job.
     *
     * @param jobId the ID of the job to retrieve the status for
     * @return a CompletableFuture that completes with the job status
```

```
     */
    public CompletableFuture<String> describeJobAsync(String jobId) {
        DescribeJobsRequest describeJobsRequest = DescribeJobsRequest.builder()
            .jobs(jobId)
            .build();

        CompletableFuture<DescribeJobsResponse> responseFuture =
getAsyncClient().describeJobs(describeJobsRequest);
        return responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
            }
        }).thenApply(response -> response.jobs().get(0).status().toString());
    }

    /**
     * Disables the specific job queue using the asynchronous Java client.
     *
     * @param jobQueueArn the Amazon Resource Name (ARN) of the job queue to wait
for
     * @return a {@link CompletableFuture} that completes when the job queue is
disabled
     */
    public CompletableFuture<Void> waitForJobQueueToBeDisabledAsync(String
jobQueueArn) {
        AtomicBoolean isDisabled = new AtomicBoolean(false);
        return CompletableFuture.runAsync(() -> {
            while (!isDisabled.get()) {
                DescribeJobQueuesRequest describeRequest =
DescribeJobQueuesRequest.builder()
                    .jobQueues(jobQueueArn)
                    .build();

                CompletableFuture<DescribeJobQueuesResponse> responseFuture =
getAsyncClient().describeJobQueues(describeRequest);
                responseFuture.whenComplete((describeResponse, ex) -> {
                    if (describeResponse != null) {
                        for (JobQueueDetail jobQueue : describeResponse.jobQueues())
{
                            if (jobQueue.jobQueueArn().equals(jobQueueArn) &&
jobQueue.state() == JQState.DISABLED) {
                                isDisabled.set(true);
                                break;
```

```
                            }
                        }
                    } else {
                        throw new RuntimeException("Error describing job queues",
ex);
                    }
                }).join();

                if (!isDisabled.get()) {
                    try {
                        logger.info("Waiting for job queue to be disabled...");
                        Thread.sleep(5000);
                    } catch (InterruptedException e) {
                        Thread.currentThread().interrupt();
                        throw new RuntimeException("Thread interrupted while waiting
for job queue to be disabled", e);
                    }
                }
            }
        }).whenComplete((result, throwable) -> {
            if (throwable != null) {
                throw new RuntimeException("Error while waiting for job queue to be
disabled", throwable);
            }
        });
    }

    public CompletableFuture<String> getJobQueueARN(String jobQueueName) {
        // Describe the job queue asynchronously
        CompletableFuture<DescribeJobQueuesResponse> describeJobQueuesFuture =
batchClient.describeJobQueues(
            DescribeJobQueuesRequest.builder()
                .jobQueues(jobQueueName)
                .build()
        );

        // Handle the asynchronous response and return the Job Queue ARN in the
CompletableFuture<String>
        CompletableFuture<String> jobQueueArnFuture = new CompletableFuture<>();
        describeJobQueuesFuture.whenComplete((response, error) -> {
            if (error != null) {
                if (error instanceof BatchException) {
                    logger.info("Batch error: " + ((BatchException)
error).awsErrorDetails().errorMessage());
```

```
                } else {
                    logger.info("Error describing job queue: " +
 error.getMessage());
                }
                jobQueueArnFuture.completeExceptionally(new RuntimeException("Failed
 to retrieve Job Queue ARN", error));
            } else {
                if (response.jobQueues().isEmpty()) {
                    jobQueueArnFuture.completeExceptionally(new
 RuntimeException("Job queue not found: " + jobQueueName));
                } else {
                    // Assuming only one job queue is returned for the given name
                    String jobQueueArn = response.jobQueues().get(0).jobQueueArn();
                    jobQueueArnFuture.complete(jobQueueArn);
                }
            }
        });

        return jobQueueArnFuture;
    }

    private static String getComputeEnvironmentName(String computeEnvironment) {
        String[] parts = computeEnvironment.split("/");
        if (parts.length == 2) {
            return parts[1];
        }
        return null;
    }

    public CompletableFuture<String> getAccountId() {
        StsAsyncClient stsAsyncClient = StsAsyncClient.builder()
            .region(Region.US_EAST_1)
            .build();

        return stsAsyncClient.getCallerIdentity()
            .thenApply(GetCallerIdentityResponse::account);
    }


}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。

- CreateComputeEnvironment
- CreateJobQueue
- DeleteComputeEnvironment
- DeleteJobQueue
- DeregisterJobDefinition
- DescribeComputeEnvironments
- DescribeJobQueues
- DescribeJobs
- ListJobsPaginator
- RegisterJobDefinition
- SubmitJob
- UpdateComputeEnvironment
- UpdateJobQueue

操作

`CreateComputeEnvironment`

以下代码示例演示了如何使用 `CreateComputeEnvironment`。

适用于 Java 的 SDK 2.x

> **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。

```
/**
 * Asynchronously creates a new compute environment in AWS Batch.
 *
 * @param computeEnvironmentName the name of the compute environment to create
 * @param batchIAMRole the IAM role to be used by the compute environment
 * @param subnet the subnet ID to be used for the compute environment
 * @param secGroup the security group ID to be used for the compute environment
```

```
    * @return a {@link CompletableFuture} representing the asynchronous operation,
which will complete with the
    *         {@link CreateComputeEnvironmentResponse} when the compute environment
has been created
    * @throws BatchException if there is an error creating the compute environment
    * @throws RuntimeException if there is an unexpected error during the operation
    */
   public CompletableFuture<CreateComputeEnvironmentResponse>
createComputeEnvironmentAsync(
       String computeEnvironmentName, String batchIAMRole, String subnet, String
secGroup) {
       CreateComputeEnvironmentRequest environmentRequest =
CreateComputeEnvironmentRequest.builder()
           .computeEnvironmentName(computeEnvironmentName)
           .type(CEType.MANAGED)
           .state(CEState.ENABLED)
           .computeResources(ComputeResource.builder()
               .type(CRType.FARGATE)
               .maxvCpus(256)
               .subnets(Collections.singletonList(subnet))
               .securityGroupIds(Collections.singletonList(secGroup))
               .build())
           .serviceRole(batchIAMRole)
           .build();

       CompletableFuture<CreateComputeEnvironmentResponse> response =
getAsyncClient().createComputeEnvironment(environmentRequest);
       response.whenComplete((resp, ex) -> {
           if (ex != null) {
              String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
              throw new RuntimeException(errorMessage, ex);
           }
       });

       return response;
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateComputeEnvironment中的。

CreateJobQueue

以下代码示例演示了如何使用 CreateJobQueue。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Creates a job queue asynchronously.
    *
    * @param jobQueueName the name of the job queue to create
    * @param computeEnvironmentName the name of the compute environment to
associate with the job queue
    * @return a CompletableFuture that completes with the Amazon Resource Name
(ARN) of the job queue
    */
   public CompletableFuture<String> createJobQueueAsync(String jobQueueName, String
computeEnvironmentName) {
       if (jobQueueName == null || jobQueueName.isEmpty()) {
           throw new IllegalArgumentException("Job queue name cannot be null or
empty");
       }
       if (computeEnvironmentName == null || computeEnvironmentName.isEmpty()) {
           throw new IllegalArgumentException("Compute environment name cannot be
null or empty");
       }

       CreateJobQueueRequest request = CreateJobQueueRequest.builder()
           .jobQueueName(jobQueueName)
           .priority(1)
           .computeEnvironmentOrder(ComputeEnvironmentOrder.builder()
               .computeEnvironment(computeEnvironmentName)
               .order(1)
               .build())
           .build();

       CompletableFuture<CreateJobQueueResponse> response =
getAsyncClient().createJobQueue(request);
```

```
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                throw new RuntimeException(errorMessage, ex);
            }
        });

        return response.thenApply(CreateJobQueueResponse::jobQueueArn);
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateJobQueue中的。

`DeleteComputeEnvironment`

以下代码示例演示了如何使用 `DeleteComputeEnvironment`。

适用于 Java 的 SDK 2.x

> **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。

```
    public CompletableFuture<DeleteComputeEnvironmentResponse>
deleteComputeEnvironmentAsync(String computeEnvironmentName) {
        DeleteComputeEnvironmentRequest deleteComputeEnvironment =
DeleteComputeEnvironmentRequest.builder()
            .computeEnvironment(computeEnvironmentName)
            .build();

        return getAsyncClient().deleteComputeEnvironment(deleteComputeEnvironment)
            .whenComplete((response, ex) -> {
                if (ex != null) {
                    Throwable cause = ex.getCause();
                    if (cause instanceof BatchException) {
                        throw new RuntimeException(cause);
                    } else {
                        throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
```

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

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteComputeEnvironment中的。

`DeleteJobQueue`

以下代码示例演示了如何使用 `DeleteJobQueue`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Deletes a Batch job queue asynchronously.
     *
     * @param jobQueueArn The Amazon Resource Name (ARN) of the job queue to delete.
     * @return A CompletableFuture that represents the asynchronous deletion of the
job queue.
     *         The future completes when the job queue has been successfully deleted
or if an error occurs.
     *         If successful, the future will be completed with a {@code Void}
value.
     *         If an error occurs, the future will be completed exceptionally with
the thrown exception.
     */
    public CompletableFuture<Void> deleteJobQueueAsync(String jobQueueArn) {
        DeleteJobQueueRequest deleteRequest = DeleteJobQueueRequest.builder()
            .jobQueue(jobQueueArn)
            .build();

        CompletableFuture<DeleteJobQueueResponse> responseFuture =
getAsyncClient().deleteJobQueue(deleteRequest);
```

```
        return responseFuture.whenComplete((deleteResponse, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete job queue: " +
ex.getMessage(), ex);
            }
        }).thenApply(deleteResponse -> null);
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteJobQueue中的。

`DeregisterJobDefinition`

以下代码示例演示了如何使用 `DeregisterJobDefinition`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Deregisters a job definition asynchronously.
     *
     * @param jobDefinition the name of the job definition to be deregistered
     * @return a CompletableFuture that completes when the job definition has been
deregistered
     * or an exception has occurred
     */
    public CompletableFuture<DeregisterJobDefinitionResponse>
deregisterJobDefinitionAsync(String jobDefinition) {
        DeregisterJobDefinitionRequest jobDefinitionRequest =
DeregisterJobDefinitionRequest.builder()
            .jobDefinition(jobDefinition)
            .build();

        CompletableFuture<DeregisterJobDefinitionResponse> responseFuture =
getAsyncClient().deregisterJobDefinition(jobDefinitionRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
```

```
                throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
            }
        });

        return responseFuture;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeregisterJobDefinition中的。

`DescribeComputeEnvironments`

以下代码示例演示了如何使用 `DescribeComputeEnvironments`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Checks the status of the specified compute environment.
     *
     * @param computeEnvironmentName the name of the compute environment to check
     * @return a CompletableFuture containing the status of the compute environment,
or "ERROR" if an exception occurs
     */
    public CompletableFuture<String> checkComputeEnvironmentsStatus(String
computeEnvironmentName) {
        if (computeEnvironmentName == null || computeEnvironmentName.isEmpty()) {
            throw new IllegalArgumentException("Compute environment name cannot be
null or empty");
        }

        DescribeComputeEnvironmentsRequest environmentsRequest =
DescribeComputeEnvironmentsRequest.builder()
            .computeEnvironments(computeEnvironmentName)
            .build();
```

```
        CompletableFuture<DescribeComputeEnvironmentsResponse> response =
getAsyncClient().describeComputeEnvironments(environmentsRequest);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                String errorMessage = "Unexpected error occurred: " +
ex.getMessage();
                throw new RuntimeException(errorMessage, ex);
            }
        });

        return response.thenApply(resp -> resp.computeEnvironments().stream()
            .map(env -> env.statusAsString())
            .findFirst()
            .orElse("UNKNOWN"));
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[DescribeComputeEnvironments](#)中的。

`DescribeJobQueues`

以下代码示例演示了如何使用 `DescribeJobQueues`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Asynchronously describes the job queue associated with the specified compute
environment.
     *
     * @param computeEnvironmentName the name of the compute environment to find the
associated job queue for
     * @return a {@link CompletableFuture} that, when completed, contains the job
queue ARN associated with the specified compute environment
     * @throws RuntimeException if the job queue description fails
```

```
     */
    public CompletableFuture<String> describeJobQueueAsync(String
computeEnvironmentName) {
        DescribeJobQueuesRequest describeJobQueuesRequest =
DescribeJobQueuesRequest.builder()
            .build();

        CompletableFuture<DescribeJobQueuesResponse> responseFuture =
getAsyncClient().describeJobQueues(describeJobQueuesRequest);
        return responseFuture.whenComplete((describeJobQueuesResponse, ex) -> {
            if (describeJobQueuesResponse != null) {
                String jobQueueARN;
                for (JobQueueDetail jobQueueDetail :
describeJobQueuesResponse.jobQueues()) {
                    for (ComputeEnvironmentOrder computeEnvironmentOrder :
jobQueueDetail.computeEnvironmentOrder()) {
                        String computeEnvironment =
computeEnvironmentOrder.computeEnvironment();
                        String name = getComputeEnvironmentName(computeEnvironment);
                        if (name.equals(computeEnvironmentName)) {
                            jobQueueARN = jobQueueDetail.jobQueueArn();
                            logger.info("Job queue ARN associated with the compute
environment: " + jobQueueARN);
                        }
                    }
                }
            } else {
                throw new RuntimeException("Failed to describe job queue: " +
ex.getMessage(), ex);
            }
        }).thenApply(describeJobQueuesResponse -> {
            String jobQueueARN = "";
            for (JobQueueDetail jobQueueDetail :
describeJobQueuesResponse.jobQueues()) {
                for (ComputeEnvironmentOrder computeEnvironmentOrder :
jobQueueDetail.computeEnvironmentOrder()) {
                    String computeEnvironment =
computeEnvironmentOrder.computeEnvironment();
                    String name = getComputeEnvironmentName(computeEnvironment);
                    if (name.equals(computeEnvironmentName)) {
                        jobQueueARN = jobQueueDetail.jobQueueArn();
                    }
                }
            }
```

```
            return jobQueueARN;
        });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeJobQueues中的。

`DescribeJobs`

以下代码示例演示了如何使用 `DescribeJobs`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Asynchronously retrieves the status of a specific job.
     *
     * @param jobId the ID of the job to retrieve the status for
     * @return a CompletableFuture that completes with the job status
     */
    public CompletableFuture<String> describeJobAsync(String jobId) {
        DescribeJobsRequest describeJobsRequest = DescribeJobsRequest.builder()
            .jobs(jobId)
            .build();

        CompletableFuture<DescribeJobsResponse> responseFuture =
getAsyncClient().describeJobs(describeJobsRequest);
        return responseFuture.whenComplete((response, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
            }
        }).thenApply(response -> response.jobs().get(0).status().toString());
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeJobs中的。

`ListJobsPaginator`

以下代码示例演示了如何使用 `ListJobsPaginator`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Asynchronously lists the jobs in the specified job queue with the given job
status.
    *
    * @param jobQueue the name of the job queue to list jobs from
    * @return a List<JobSummary> that contains the jobs that succeeded
    */
   public List<JobSummary> listJobsAsync(String jobQueue) {
       if (jobQueue == null || jobQueue.isEmpty()) {
           throw new IllegalArgumentException("Job queue cannot be null or empty");
       }

       ListJobsRequest listJobsRequest = ListJobsRequest.builder()
           .jobQueue(jobQueue)
           .jobStatus(JobStatus.SUCCEEDED)  // Filter jobs by status.
           .build();

       List<JobSummary> jobSummaries = new ArrayList<>();
       ListJobsPublisher listJobsPaginator =
getAsyncClient().listJobsPaginator(listJobsRequest);
       CompletableFuture<Void> future = listJobsPaginator.subscribe(response -> {
           jobSummaries.addAll(response.jobSummaryList());
       });
       future.join();
       return jobSummaries;
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListJobsPaginator中的。

`RegisterJobDefinition`

以下代码示例演示了如何使用 `RegisterJobDefinition`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Registers a new job definition asynchronously in AWS Batch.
    * <p>
    * When using Fargate as the compute environment, it is crucial to set the
    * {@link NetworkConfiguration} with {@link AssignPublicIp#ENABLED} to
    * ensure proper networking configuration for the Fargate tasks. This
    * allows the tasks to communicate with external services, access the
    * internet, or communicate within a VPC.
    *
    * @param jobDefinitionName the name of the job definition to be registered
    * @param executionRoleARN the ARN (Amazon Resource Name) of the execution role
    *                         that provides permissions for the containers in the
job
    * @param cpuArch a value of either X86_64 or ARM64 required for the service
call
    * @return a CompletableFuture that completes with the ARN of the registered
    *         job definition upon successful execution, or completes exceptionally
with
    *         an error if the registration fails
    */
   public CompletableFuture<String> registerJobDefinitionAsync(String
jobDefinitionName, String executionRoleARN, String image, String cpuArch) {
       NetworkConfiguration networkConfiguration = NetworkConfiguration.builder()
               .assignPublicIp(AssignPublicIp.ENABLED)
               .build();

       ContainerProperties containerProperties = ContainerProperties.builder()
               .image(image)
```

```
                .executionRoleArn(executionRoleARN)
                .resourceRequirements(
                        Arrays.asList(
                                ResourceRequirement.builder()
                                        .type(ResourceType.VCPU)
                                        .value("1")
                                        .build(),
                                ResourceRequirement.builder()
                                        .type(ResourceType.MEMORY)
                                        .value("2048")
                                        .build()
                        )
                )
                .networkConfiguration(networkConfiguration)
               .runtimePlatform(b -> b
                        .cpuArchitecture(cpuArch)
                        .operatingSystemFamily("LINUX"))
                .build();

        RegisterJobDefinitionRequest request =
RegisterJobDefinitionRequest.builder()
                .jobDefinitionName(jobDefinitionName)
                .type(JobDefinitionType.CONTAINER)
                .containerProperties(containerProperties)
                .platformCapabilities(PlatformCapability.FARGATE)
                .build();

        CompletableFuture<String> future = new CompletableFuture<>();
        getAsyncClient().registerJobDefinition(request)
                .thenApply(RegisterJobDefinitionResponse::jobDefinitionArn)
                .whenComplete((result, ex) -> {
                    if (ex != null) {
                        future.completeExceptionally(ex);
                    } else {
                        future.complete(result);
                    }
                });

        return future;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考RegisterJobDefinition中的。

SubmitJob

以下代码示例演示了如何使用 SubmitJob。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Submits a job asynchronously to the AWS Batch service.
    *
    * @param jobDefinitionName the name of the job definition to use
    * @param jobQueueName the name of the job queue to submit the job to
    * @param jobARN the Amazon Resource Name (ARN) of the job definition
    * @return a CompletableFuture that, when completed, contains the job ID of the
submitted job
    */
   public CompletableFuture<String> submitJobAsync(String jobDefinitionName, String
jobQueueName, String jobARN) {
       SubmitJobRequest jobRequest = SubmitJobRequest.builder()
           .jobDefinition(jobARN)
           .jobName(jobDefinitionName)
           .jobQueue(jobQueueName)
           .build();

       CompletableFuture<SubmitJobResponse> responseFuture =
getAsyncClient().submitJob(jobRequest);
       responseFuture.whenComplete((response, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Unexpected error occurred: " +
ex.getMessage(), ex);
           }
       });

       return responseFuture.thenApply(SubmitJobResponse::jobId);
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SubmitJob中的。

`UpdateComputeEnvironment`

以下代码示例演示了如何使用 `UpdateComputeEnvironment`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Disables the specified compute environment asynchronously.
    *
    * @param computeEnvironmentName the name of the compute environment to disable
    * @return a CompletableFuture that completes when the compute environment is
disabled
    */
   public CompletableFuture<UpdateComputeEnvironmentResponse>
disableComputeEnvironmentAsync(String computeEnvironmentName) {
       UpdateComputeEnvironmentRequest updateRequest =
UpdateComputeEnvironmentRequest.builder()
           .computeEnvironment(computeEnvironmentName)
           .state(CEState.DISABLED)
           .build();

       CompletableFuture<UpdateComputeEnvironmentResponse> responseFuture =
getAsyncClient().updateComputeEnvironment(updateRequest);
       responseFuture.whenComplete((response, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to disable compute environment: "
+ ex.getMessage(), ex);
           }
       });

       return responseFuture;
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考UpdateComputeEnvironment中的。

`UpdateJobQueue`

以下代码示例演示了如何使用 `UpdateJobQueue`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Disables the specified job queue asynchronously.
    *
    * @param jobQueueArn the Amazon Resource Name (ARN) of the job queue to be
disabled
    * @return a {@link CompletableFuture} that completes when the job queue update
operation is complete,
    *         or completes exceptionally if an error occurs during the operation
    */
   public CompletableFuture<Void> disableJobQueueAsync(String jobQueueArn) {
       UpdateJobQueueRequest updateRequest = UpdateJobQueueRequest.builder()
           .jobQueue(jobQueueArn)
           .state(JQState.DISABLED)
           .build();

       CompletableFuture<UpdateJobQueueResponse> responseFuture =
getAsyncClient().updateJobQueue(updateRequest);
       return responseFuture.whenComplete((updateResponse, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to update job queue: " +
ex.getMessage(), ex);
           }
       }).thenApply(updateResponse -> null);
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考UpdateJobQueue中的。

使用 SDK for 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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从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`

以下代码示例演示了如何使用 `GetFoundationModel`。

适用于 Java 的 SDK 2.x

> **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。

获取有关使用同步 Amazon Bedrock 客户端的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);
        }
    }
```

获取有关使用异步 Amazon Bedrock 客户端的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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[GetFoundationModel](#)中的。

`ListFoundationModels`

以下代码示例演示了如何使用 `ListFoundationModels`。

适用于 Java 的 SDK 2.x

> **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。

列出使用同步 Amazon Bedrock 客户端的 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);
        }
    }
```

列出使用异步 Amazon Bedrock 客户端的 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListFoundationModels中的。

使用 SDK for Java 2.x 的 Amazon Bedrock 运行时系统示例

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。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 场景
- Amazon Nova
- Amazon Nova Canvas
- Amazon Titan 图像生成器
- Amazon Titan 文本嵌入
- Anthropic Claude

- Cohere Command
- Meta Llama
- Mistral AI
- Stable Diffusion

场景

创建用于与 Amazon Bedrock 基础模型进行交互的平台应用程序

以下代码示例演示如何创建操场，以通过不同模态与 Amazon Bedrock 基础模型交互。

适用于 Java 的 SDK 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

使用 Amazon Bedrock 根据文本提示生成视频

以下代码示例演示如何创建一个 Spring Boot 应用程序，该应用程序使用 Amazon Bedrock 和 Nova-Reel 模型根据文本提示生成视频。

适用于 Java 的 SDK 2.x

> **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。

使用 Amazon Bedrock 和 Nova-Reel 根据文本提示生成视频。

```
import org.springframework.stereotype.Service;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

import java.util.concurrent.CompletableFuture;

@Service
public class VideoGenerationService {

    public GenerateVideoResponse generateVideo(String prompt) {

        // add S3 bucket you want to store your generated videos
        String s3Bucket = "s3://mygeneratedvidoenovatest";


        //Create json request as an instance of Document class
        Document novaRequest = prepareDocument(prompt);

        // Create request
        StartAsyncInvokeRequest request = StartAsyncInvokeRequest.builder()
                .modelId("amazon.nova-reel-v1:0")
                .modelInput(novaRequest)
                .outputDataConfig(AsyncInvokeOutputDataConfig.builder()

 .s3OutputDataConfig(AsyncInvokeS3OutputDataConfig.builder().s3Uri(s3Bucket).build())
                        .build())
                .build();

        try (BedrockRuntimeAsyncClient bedrockClient =
getBedrockRuntimeAsyncClient()) {
```

```
            CompletableFuture<StartAsyncInvokeResponse>
startAsyncInvokeResponseCompletableFuture =
bedrockClient.startAsyncInvoke(request);

            //blocking operation to wait for the AWS API response
            StartAsyncInvokeResponse startAsyncInvokeResponse =
startAsyncInvokeResponseCompletableFuture.get();
            System.out.println("invocation ARN: " +
startAsyncInvokeResponse.invocationArn());

            GenerateVideoResponse response = new GenerateVideoResponse();
            response.setStatus("inProgress");
            response.setExecutionArn(startAsyncInvokeResponse.invocationArn());

            return response;
        } catch (Exception e) {
            System.out.println(e);
            throw new RuntimeException(e);
        }

    }

    public GenerateVideoResponse checkGenerationStatus(String invocationArn) {
        GenerateVideoResponse response = new GenerateVideoResponse();

        try (BedrockRuntimeAsyncClient bedrockClient =
getBedrockRuntimeAsyncClient()) {
            //creating async request to fetch status by invocation Arn
            GetAsyncInvokeRequest asyncRequest =
GetAsyncInvokeRequest.builder().invocationArn(invocationArn).build();

            CompletableFuture<GetAsyncInvokeResponse> asyncInvoke =
bedrockClient.getAsyncInvoke(asyncRequest);

            //blocking operation to wait for the AWS API response
            GetAsyncInvokeResponse asyncInvokeResponse = asyncInvoke.get();
            System.out.println("Invocation status =" +
asyncInvokeResponse.statusAsString());

            response.setExecutionArn(invocationArn);
            response.setStatus(asyncInvokeResponse.statusAsString());
            return response;
        } catch (Exception e) {
            e.printStackTrace();
```

```
            throw new RuntimeException(e);
        }

    }

    private static BedrockRuntimeAsyncClient getBedrockRuntimeAsyncClient() {
        BedrockRuntimeAsyncClient bedrockClient =
 BedrockRuntimeAsyncClient.builder()
                .region(Region.US_EAST_1)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();
        return bedrockClient;
    }

    private static Document prepareDocument(String prompt) {
        Document textToVideoParams = Document.mapBuilder()
                .putString("text", prompt)
                .build();

        Document videoGenerationConfig = Document.mapBuilder()
                .putNumber("durationSeconds", 6)
                .putNumber("fps", 24)
                .putString("dimension", "1280x720")
                .build();

        Document novaRequest = Document.mapBuilder()
                .putString("taskType", "TEXT_VIDEO")
                .putDocument("textToVideoParams", textToVideoParams)
                .putDocument("videoGenerationConfig", videoGenerationConfig)
                .build();
        return novaRequest;
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - GetAsyncInvoke
  - StartAsyncInvoke

将工具与 Converse API 结合使用

以下代码示例展示了如何在应用程序、生成式 AI 模型和互联工具之间建立典型的交互，或者 APIs 如何调解 AI 与外界之间的交互。该代码示例以将外部天气 API 连接到人工智能模型模型为例，它可以根据用户输入提供实时天气信息。

适用于 Java 的 SDK 2.x

> **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。

该场景流程的主要执行。该场景编排了用户、Amazon Bedrock Converse API 和天气工具之间的对话。

```
/*
 This demo illustrates a tool use scenario using Amazon Bedrock's Converse API and a
 weather tool.
 The program interacts with a foundation model on Amazon Bedrock to provide weather
 information based on user
 input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.
 */
public class BedrockScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static String modelId = "amazon.nova-lite-v1:0";
    private static String defaultPrompt = "What is the weather like in Seattle?";
    private static WeatherTool weatherTool = new WeatherTool();

    // The maximum number of recursive calls allowed in the tool use function.
    // This helps prevent infinite loops and potential performance issues.
    private static int maxRecursions = 5;
    static BedrockActions bedrockActions = new BedrockActions();
    public static boolean interactive = true;

    private static final String systemPrompt = """
            You are a weather assistant that provides current weather data for user-
specified locations using only
            the Weather_Tool, which expects latitude and longitude. Infer the
 coordinates from the location yourself.
```

```
            If the user provides coordinates, infer the approximate location and
 refer to it in your response.
            To use the tool, you strictly apply the provided tool specification.

            - Explain your step-by-step process, and give brief updates before each
 step.
            - Only use the Weather_Tool for data. Never guess or make up
 information.
            - Repeat the tool use for subsequent requests if necessary.
            - If the tool errors, apologize, explain weather is unavailable, and
 suggest other options.
            - Report temperatures in °C (°F) and wind in km/h (mph). Keep weather
 reports concise. Sparingly use
              emojis where appropriate.
            - Only respond to weather queries. Remind off-topic users of your
 purpose.
            - Never claim to search online, access external data, or use tools
 besides Weather_Tool.
            - Complete the entire process until you have all required data before
 sending the complete response.
            """;

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("""
                ==================================================
                Welcome to the Amazon Bedrock Tool Use demo!
                ==================================================

                This assistant provides current weather information for user-
specified locations.
                You can ask for weather details by providing the location name or
 coordinates.

                Example queries:
                - What's the weather like in New York?
                - Current weather for latitude 40.70, longitude -74.01
                - Is it warmer in Rome or Barcelona today?

                To exit the program, simply type 'x' and press Enter.

                P.S.: You're not limited to single locations, or even to using
 English!
                Have fun and experiment with the app!
```

```
                """);
        System.out.println(DASHES);

        try {
            runConversation(scanner);

        } catch (Exception ex) {
            System.out.println("There was a problem running the scenario: " +
ex.getMessage());
        }

        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("Amazon Bedrock Converse API with Tool Use Feature
Scenario is complete.");
        System.out.println(DASHES);
    }

    /**
     * Starts the conversation with the user and handles the interaction with
Bedrock.
     */
    private static List<Message> runConversation(Scanner scanner) {
        List<Message> conversation = new ArrayList<>();

        // Get the first user input
        String userInput = getUserInput("Your weather info request:", scanner);
        System.out.println(userInput);

        while (userInput != null) {
            ContentBlock block = ContentBlock.builder()
                    .text(userInput)
                    .build();

            List<ContentBlock> blockList = new ArrayList<>();
            blockList.add(block);

            Message message = Message.builder()
                    .role(ConversationRole.USER)
                    .content(blockList)
                    .build();

            conversation.add(message);
```

```
            // Send the conversation to Amazon Bedrock.
            ConverseResponse bedrockResponse =
sendConversationToBedrock(conversation);

            // Recursively handle the model's response until the model has returned
its final response or the recursion counter has reached 0.
            processModelResponse(bedrockResponse, conversation, maxRecursions);

            // Repeat the loop until the user decides to exit the application.
            userInput = getUserInput("Your weather info request:", scanner);
        }
        printFooter();
        return conversation;
    }

    /**
     * Processes the response from the model and updates the conversation
accordingly.
     *
     * @param modelResponse the response from the model
     * @param conversation  the ongoing conversation
     * @param maxRecursion  the maximum number of recursions allowed
     */
    private static void processModelResponse(ConverseResponse modelResponse,
List<Message> conversation, int maxRecursion) {
        if (maxRecursion <= 0) {
            // Stop the process, the number of recursive calls could indicate an
infinite loop
            System.out.println("\tWarning: Maximum number of recursions reached.
Please try again.");
        }

        // Append the model's response to the ongoing conversation
        conversation.add(modelResponse.output().message());

        String modelResponseVal = modelResponse.stopReasonAsString();
        if (modelResponseVal.compareTo("tool_use") == 0) {
            // If the stop reason is "tool_use", forward everything to the tool use
handler
            handleToolUse(modelResponse.output(), conversation, maxRecursion - 1);
        }

        if (modelResponseVal.compareTo("end_turn") == 0) {
```

```
            // If the stop reason is "end_turn", print the model's response text,
and finish the process

PrintModelResponse(modelResponse.output().message().content().get(0).text());
            if (!interactive) {
                defaultPrompt = "x";
            }
        }
    }

    /**
     * Handles the use of a tool by the model in a conversation.
     *
     * @param modelResponse the response from the model, which may include a tool
use request
     * @param conversation  the current conversation, which will be updated with the
tool use results
     * @param maxRecursion  the maximum number of recursive calls allowed to handle
the model's response
     */
    private static void handleToolUse(ConverseOutput modelResponse, List<Message>
conversation, int maxRecursion) {
        List<ContentBlock> toolResults = new ArrayList<>();

        // The model's response can consist of multiple content blocks
        for (ContentBlock contentBlock : modelResponse.message().content()) {
            if (contentBlock.text() != null && !contentBlock.text().isEmpty()) {
                // If the content block contains text, print it to the console
                PrintModelResponse(contentBlock.text());
            }

            if (contentBlock.toolUse() != null) {
                ToolResponse toolResponse = invokeTool(contentBlock.toolUse());

                // Add the tool use ID and the tool's response to the list of
results
                List<ToolResultContentBlock> contentBlockList = new ArrayList<>();
                ToolResultContentBlock block = ToolResultContentBlock.builder()
                        .json(toolResponse.getContent())
                        .build();
                contentBlockList.add(block);

                ToolResultBlock toolResultBlock = ToolResultBlock.builder()
                        .toolUseId(toolResponse.getToolUseId())
```

```
                        .content(contentBlockList)
                        .build();

                ContentBlock contentBlock1 = ContentBlock.builder()
                        .toolResult(toolResultBlock)
                        .build();

                toolResults.add(contentBlock1);
            }
        }

        // Embed the tool results in a new user message
        Message message = Message.builder()
                .role(ConversationRole.USER)
                .content(toolResults)
                .build();

        // Append the new message to the ongoing conversation
        //conversation.add(message);
        conversation.add(message);

        // Send the conversation to Amazon Bedrock
        var response = sendConversationToBedrock(conversation);

        // Recursively handle the model's response until the model has returned its
final response or the recursion counter has reached 0
        processModelResponse(response, conversation, maxRecursion);
    }

    // Invokes the specified tool with the given payload and returns the tool's
response.
    // If the requested tool does not exist, an error message is returned.
    private static ToolResponse invokeTool(ToolUseBlock payload) {
        String toolName = payload.name();

        if (Objects.equals(toolName, "Weather_Tool")) {
            Map<String, Document> inputData = payload.input().asMap();
            printToolUse(toolName, inputData);

            // Invoke the weather tool with the input data provided
            Document weatherResponse =
weatherTool.fetchWeatherData(inputData.get("latitude").toString(),
inputData.get("longitude").toString());
```

```
            ToolResponse toolResponse = new ToolResponse();
            toolResponse.setContent(weatherResponse);
            toolResponse.setToolUseId(payload.toolUseId());
            return toolResponse;
        } else {
            String errorMessage = "The requested tool with name " + toolName + "
does not exist.";
            System.out.println(errorMessage);
            return null;
        }
    }

    public static void printToolUse(String toolName, Map<String, Document>
inputData) {
        System.out.println("Invoking tool: " + toolName + " with input: " +
inputData.get("latitude").toString() + ", " + inputData.get("longitude").toString()
+ "...");
    }

    private static void PrintModelResponse(String message) {
        System.out.println("\tThe model's response:\n");
        System.out.println(message);
        System.out.println("");
    }

    private static ConverseResponse sendConversationToBedrock(List<Message>
conversation) {
        System.out.println("Calling Bedrock...");

        try {
            return bedrockActions.sendConverseRequestAsync(modelId, systemPrompt,
conversation, weatherTool.getToolSpec());
        } catch (ModelNotReadyException ex) {
             System.err.println("Model is not ready. Please try again later: " +
ex.getMessage());
            throw ex;
        } catch (BedrockRuntimeException ex) {
            System.err.println("Bedrock service error: " + ex.getMessage());
            throw ex;
        } catch (RuntimeException ex) {
            System.err.println("Unexpected error occurred: " + ex.getMessage());
            throw ex;
        }
    }
```

```
   private static ConverseResponse sendConversationToBedrockwithSpec(List<Message>
conversation, ToolSpecification toolSpec) {
       System.out.println("Calling Bedrock...");

       // Send the conversation, system prompt, and tool configuration, and return
the response
       return bedrockActions.sendConverseRequestAsync(modelId, systemPrompt,
conversation, toolSpec);
   }

   public static String getUserInput(String prompt, Scanner scanner) {
       String userInput = defaultPrompt;
       if (interactive) {
           System.out.println("*".repeat(80));
           System.out.println(prompt + " (x to exit): \n\t");
           userInput = scanner.nextLine();
       }

       if (userInput == null || userInput.trim().isEmpty()) {
           return getUserInput("\tPlease enter your weather info request, e.g., the
name of a city", scanner);
       }

       if (userInput.equalsIgnoreCase("x")) {
           return null;
       }

       return userInput;
   }

   private static void waitForInputToContinue(Scanner scanner) {
       while (true) {
           System.out.println("");
           System.out.println("Enter 'c' followed by <ENTER> to continue:");
           String input = scanner.nextLine();

           if (input.trim().equalsIgnoreCase("c")) {
               System.out.println("Continuing with the program...");
               System.out.println("");
               break;
           } else {
               // Handle invalid input.
               System.out.println("Invalid input. Please try again.");
```

```
            }
        }
    }

    public static void printFooter() {
        System.out.println("""
                ================================================
                Thank you for checking out the Amazon Bedrock Tool Use demo. We hope
 you
                learned something new, or got some inspiration for your own apps
 today!

                For more Bedrock examples in different programming languages, have a
 look at:
                https://docs.aws.amazon.com/bedrock/latest/userguide/
service_code_examples.html
                ================================================
                """);
    }
}
```

演示使用的天气工具。该文件定义了工具规范，并实现了从 Open-Meteo API 中检索天气数据的逻辑。

```
public class WeatherTool {

    private static final Logger logger = LoggerFactory.getLogger(WeatherTool.class);
    private static java.net.http.HttpClient httpClient = null;

    /**
     * Returns the JSON Schema specification for the Weather tool. The tool
 specification
     * defines the input schema and describes the tool's functionality.
     * For more information, see https://json-schema.org/understanding-json-schema/
reference.
     *
     * @return The tool specification for the Weather tool.
     */
    public ToolSpecification getToolSpec() {
        Map<String, Document> latitudeMap = new HashMap<>();
        latitudeMap.put("type", Document.fromString("string"));
```

```
        latitudeMap.put("description", Document.fromString("Geographical WGS84
latitude of the location."));

        // Create the nested "longitude" object
        Map<String, Document> longitudeMap = new HashMap<>();
        longitudeMap.put("type", Document.fromString("string"));
        longitudeMap.put("description", Document.fromString("Geographical WGS84
longitude of the location."));

        // Create the "properties" object
        Map<String, Document> propertiesMap = new HashMap<>();
        propertiesMap.put("latitude", Document.fromMap(latitudeMap));
        propertiesMap.put("longitude", Document.fromMap(longitudeMap));

        // Create the "required" array
        List<Document> requiredList = new ArrayList<>();
        requiredList.add(Document.fromString("latitude"));
        requiredList.add(Document.fromString("longitude"));

        // Create the root object
        Map<String, Document> rootMap = new HashMap<>();
        rootMap.put("type", Document.fromString("object"));
        rootMap.put("properties", Document.fromMap(propertiesMap));
        rootMap.put("required", Document.fromList(requiredList));

        // Now create the Document representing the JSON schema
        Document document = Document.fromMap(rootMap);

        ToolSpecification specification = ToolSpecification.builder()
            .name("Weather_Tool")
            .description("Get the current weather for a given location, based on its
WGS84 coordinates.")
            .inputSchema(ToolInputSchema.builder()
                .json(document)
                .build())
            .build();

        return specification;
    }

    /**
     * Fetches weather data for the given latitude and longitude.
     *
     * @param latitude  the latitude coordinate
```

```
     * @param longitude the longitude coordinate
     * @return a {@link CompletableFuture} containing the weather data as a JSON
 string
     */
    public Document fetchWeatherData(String latitude, String longitude) {
        HttpClient httpClient = HttpClient.newHttpClient();

        // Ensure no extra double quotes
        latitude = latitude.replace("\"", "");
        longitude = longitude.replace("\"", "");

        String endpoint = "https://api.open-meteo.com/v1/forecast";
        String url = String.format("%s?latitude=%s&longitude=
%s&current_weather=True", endpoint, latitude, longitude);

        HttpRequest request = HttpRequest.newBuilder()
            .uri(URI.create(url))
            .build();

        try {
            HttpResponse<String> response = httpClient.send(request,
 HttpResponse.BodyHandlers.ofString());
            if (response.statusCode() == 200) {
                String weatherJson = response.body();
                System.out.println(weatherJson);
                ObjectMapper objectMapper = new ObjectMapper();
                Map<String, Object> rawMap = objectMapper.readValue(weatherJson, new
 TypeReference<Map<String, Object>>() {});
                Map<String, Document> documentMap = convertToDocumentMap(rawMap);


                Document weatherDocument = Document.fromMap(documentMap);
                System.out.println(weatherDocument);
                return weatherDocument;
            } else {
                throw new RuntimeException("Error fetching weather data: " +
 response.statusCode());
            }
        } catch (Exception e) {
            System.out.println("Error fetching weather data: " + e.getMessage());
            throw new RuntimeException("Error fetching weather data", e);
        }

    }
```

```
    private static Map<String, Document> convertToDocumentMap(Map<String, Object>
 inputMap) {
        Map<String, Document> result = new HashMap<>();
        for (Map.Entry<String, Object> entry : inputMap.entrySet()) {
            result.put(entry.getKey(), convertToDocument(entry.getValue()));
        }
        return result;
    }

    // Convert different types of Objects to Document
    private static Document convertToDocument(Object value) {
        if (value instanceof Map) {
            return Document.fromMap(convertToDocumentMap((Map<String, Object>)
 value));
        } else if (value instanceof Integer) {
            return Document.fromNumber(SdkNumber.fromInteger((Integer) value));
        } else if (value instanceof Double) {  //
            return Document.fromNumber(SdkNumber.fromDouble((Double) value));
        } else if (value instanceof Boolean) {
            return Document.fromBoolean((Boolean) value);
        } else if (value instanceof String) {
            return Document.fromString((String) value);
        }
        return Document.fromNull(); // Handle null values safely
    }
}
```

Converse API 操作和工具配置。

```
    /**
     * Sends an asynchronous converse request to the AI model.
     *
     * @param modelId      the unique identifier of the AI model to be used for the
 converse request
     * @param systemPrompt the system prompt to be included in the converse request
     * @param conversation a list of messages representing the conversation history
     * @param toolSpec     the specification of the tool to be used in the converse
 request
     * @return the converse response received from the AI model
     */
```

```
    public ConverseResponse sendConverseRequestAsync(String modelId, String
systemPrompt, List<Message> conversation, ToolSpecification toolSpec) {
        List<Tool> toolList = new ArrayList<>();
        Tool tool = Tool.builder()
            .toolSpec(toolSpec)
            .build();

        toolList.add(tool);

        ToolConfiguration configuration = ToolConfiguration.builder()
            .tools(toolList)
            .build();

        SystemContentBlock block = SystemContentBlock.builder()
            .text(systemPrompt)
            .build();

        ConverseRequest request = ConverseRequest.builder()
            .modelId(modelId)
            .system(block)
            .messages(conversation)
            .toolConfig(configuration)
            .build();

        try {
            ConverseResponse response = getClient().converse(request).join();
            return response;

        } catch (ModelNotReadyException ex) {
            throw new RuntimeException("Model is not ready: " + ex.getMessage(),
ex);
        } catch (BedrockRuntimeException ex) {
            throw new RuntimeException("Failed to converse with Bedrock model: " +
ex.getMessage(), ex);
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 Converse。

Amazon Nova

Converse

以下代码示例演示如何使用 Bedrock 的 Converse API 向 Amazon Nova 发送文本消息。

适用于 Java 的 SDK 2.x

> **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。

使用 Bedrock 的 Converse API 和异步 Java 客户端向 Amazon Nova 发送文本消息。

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

import java.util.concurrent.CompletableFuture;

/**
 * This example demonstrates how to use the Amazon Nova foundation models
 * with an asynchronous Amazon Bedrock runtime client to generate text.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure and send a request
 * - Process the response
 */
public class ConverseAsync {

    public static String converseAsync() {

        // Step 1: Create the Amazon Bedrock runtime client
        // The runtime client handles the communication with AI models on Amazon
 Bedrock
        BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
```

```
                .build();

        // Step 2: Specify which model to use
        // Available Amazon Nova models and their characteristics:
        // - Amazon Nova Micro: Text-only model optimized for lowest latency and
 cost
        // - Amazon Nova Lite:  Fast, low-cost multimodal model for image, video,
 and text
        // - Amazon Nova Pro:   Advanced multimodal model balancing accuracy, speed,
 and cost
        //
        // For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        String modelId = "amazon.nova-lite-v1:0";

        // Step 3: Create the message
        // The message includes the text prompt and specifies that it comes from the
 user
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Step 4: Configure the request
        // Optional parameters to control the model's response:
        // - maxTokens: maximum number of tokens to generate
        // - temperature: randomness (max: 1.0, default: 0.7)
        //   OR
        // - topP: diversity of word choice (max: 1.0, default: 0.9)
        // Note: Use either temperature OR topP, but not both
        ConverseRequest request = ConverseRequest.builder()
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                                .maxTokens(500)     // The maximum response length
                                .temperature(0.5F)  // Using temperature for
 randomness control
                        //.topP(0.9F)       // Alternative: use topP instead of
 temperature
                ).build();
```

```
        // Step 5: Send and process the request asynchronously
        // - Send the request to the model
        // - Extract and return the generated text from the response
        try {
            CompletableFuture<ConverseResponse> asyncResponse =
 client.converse(request);
            return asyncResponse.thenApply(
                    response -> response.output().message().content().get(0).text()
            ).get();

        } catch (Exception e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        String response = converseAsync();
        System.out.println(response);
    }
}
```

使用 Bedrock 的 Converse API 向 Amazon Nova 发送文本消息。

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

/**
 * This example demonstrates how to use the Amazon Nova foundation models
 * with a synchronous Amazon Bedrock runtime client to generate text.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure and send a request
 * - Process the response
 */
public class Converse {
```

```
    public static String converse() {

        // Step 1: Create the Amazon Bedrock runtime client
        // The runtime client handles the communication with AI models on Amazon
 Bedrock
        BedrockRuntimeClient client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Step 2: Specify which model to use
        // Available Amazon Nova models and their characteristics:
        // - Amazon Nova Micro: Text-only model optimized for lowest latency and
 cost
        // - Amazon Nova Lite:  Fast, low-cost multimodal model for image, video,
 and text
        // - Amazon Nova Pro:   Advanced multimodal model balancing accuracy, speed,
 and cost
        //
        // For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        String modelId = "amazon.nova-lite-v1:0";

        // Step 3: Create the message
        // The message includes the text prompt and specifies that it comes from the
 user
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Step 4: Configure the request
        // Optional parameters to control the model's response:
        // - maxTokens: maximum number of tokens to generate
        // - temperature: randomness (max: 1.0, default: 0.7)
        //   OR
        // - topP: diversity of word choice (max: 1.0, default: 0.9)
        // Note: Use either temperature OR topP, but not both
        ConverseRequest request = ConverseRequest.builder()
                .modelId(modelId)
```

```
                .messages(message)
                .inferenceConfig(config -> config
                                .maxTokens(500)     // The maximum response length
                                .temperature(0.5F)  // Using temperature for
 randomness control
                        //.topP(0.9F)       // Alternative: use topP instead of
 temperature
                ).build();

        // Step 5: Send and process the request
        // - Send the request to the model
        // - Extract and return the generated text from the response
        try {
            ConverseResponse response = client.converse(request);
            return response.output().message().content().get(0).text();

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        String response = converse();
        System.out.println(response);
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 Converse。

ConverseStream

以下代码示例演示如何使用 Bedrock 的 Converse API 向 Amazon Nova 发送文本消息并实时处理响应流。

适用于 Java 的 SDK 2.x

> **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。

使用 Bedrock 的 Converse API 向 Amazon Nova 发送文本消息并实时处理回复流。

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

import java.util.concurrent.ExecutionException;

/**
 * This example demonstrates how to use the Amazon Nova foundation models with an
 * asynchronous Amazon Bedrock runtime client to generate streaming text responses.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure a streaming request
 * - Set up a stream handler to process the response chunks
 * - Process the streaming response
 */
public class ConverseStream {

    public static void converseStream() {

        // Step 1: Create the Amazon Bedrock runtime client
        // The runtime client handles the communication with AI models on Amazon
 Bedrock
        BedrockRuntimeAsyncClient client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Step 2: Specify which model to use
        // Available Amazon Nova models and their characteristics:
```

```
        // - Amazon Nova Micro: Text-only model optimized for lowest latency and
 cost
        // - Amazon Nova Lite:  Fast, low-cost multimodal model for image, video,
 and text
        // - Amazon Nova Pro:   Advanced multimodal model balancing accuracy, speed,
 and cost
        //
        // For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        String modelId = "amazon.nova-lite-v1:0";

        // Step 3: Create the message
        // The message includes the text prompt and specifies that it comes from the
 user
        var inputText = "Describe the purpose of a 'hello world' program in one
 paragraph";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Step 4: Configure the request
        // Optional parameters to control the model's response:
        // - maxTokens: maximum number of tokens to generate
        // - temperature: randomness (max: 1.0, default: 0.7)
        //   OR
        // - topP: diversity of word choice (max: 1.0, default: 0.9)
        // Note: Use either temperature OR topP, but not both
        ConverseStreamRequest request = ConverseStreamRequest.builder()
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                                .maxTokens(500)     // The maximum response length
                                .temperature(0.5F)  // Using temperature for
 randomness control
                        //.topP(0.9F)       // Alternative: use topP instead of
 temperature
                ).build();

        // Step 5: Set up the stream handler
        // The stream handler processes chunks of the response as they arrive
        // - onContentBlockDelta: Processes each text chunk
        // - onError: Handles any errors during streaming
```

```
        var streamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            System.out.print(chunk.delta().text());
                            System.out.flush();  // Ensure immediate output of each
 chunk
                        }).build())
                .onError(err -> System.err.printf("Can't invoke '%s': %s", modelId,
 err.getMessage()))
                .build();

        // Step 6: Send the streaming request and process the response
        // - Send the request to the model
        // - Attach the handler to process response chunks as they arrive
        // - Handle any errors during streaming
        try {
            client.converseStream(request, streamHandler).get();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
        }
    }

    public static void main(String[] args) {
        converseStream();
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ConverseStream中的。

场景：将工具与 Converse API 搭配使用

以下代码示例展示了如何在应用程序、生成式 AI 模型和互联工具之间建立典型的交互，或者 APIs 如何调解 AI 与外界之间的交互。该代码示例以将外部天气 API 连接到人工智能模型模型为例，它可以根据用户输入提供实时天气信息。

适用于 Java 的 SDK 2.x

> **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。

该场景流程的主要执行。该场景编排了用户、Amazon Bedrock Converse API 和天气工具之间的对话。

```
/*
 This demo illustrates a tool use scenario using Amazon Bedrock's Converse API and a
 weather tool.
 The program interacts with a foundation model on Amazon Bedrock to provide weather
 information based on user
 input. It uses the Open-Meteo API (https://open-meteo.com) to retrieve current
 weather data for a given location.
 */
public class BedrockScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static String modelId = "amazon.nova-lite-v1:0";
    private static String defaultPrompt = "What is the weather like in Seattle?";
    private static WeatherTool weatherTool = new WeatherTool();

    // The maximum number of recursive calls allowed in the tool use function.
    // This helps prevent infinite loops and potential performance issues.
    private static int maxRecursions = 5;
    static BedrockActions bedrockActions = new BedrockActions();
    public static boolean interactive = true;

    private static final String systemPrompt = """
            You are a weather assistant that provides current weather data for user-
specified locations using only
            the Weather_Tool, which expects latitude and longitude. Infer the
 coordinates from the location yourself.
            If the user provides coordinates, infer the approximate location and
 refer to it in your response.
            To use the tool, you strictly apply the provided tool specification.

            - Explain your step-by-step process, and give brief updates before each
 step.
```

```
            - Only use the Weather_Tool for data. Never guess or make up
 information.
            - Repeat the tool use for subsequent requests if necessary.
            - If the tool errors, apologize, explain weather is unavailable, and
 suggest other options.
            - Report temperatures in °C (°F) and wind in km/h (mph). Keep weather
 reports concise. Sparingly use
              emojis where appropriate.
            - Only respond to weather queries. Remind off-topic users of your
 purpose.
            - Never claim to search online, access external data, or use tools
 besides Weather_Tool.
            - Complete the entire process until you have all required data before
 sending the complete response.
            """;

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("""
                ==================================================
                Welcome to the Amazon Bedrock Tool Use demo!
                ==================================================

                This assistant provides current weather information for user-
specified locations.
                You can ask for weather details by providing the location name or
 coordinates.

                Example queries:
                - What's the weather like in New York?
                - Current weather for latitude 40.70, longitude -74.01
                - Is it warmer in Rome or Barcelona today?

                To exit the program, simply type 'x' and press Enter.

                P.S.: You're not limited to single locations, or even to using
 English!
                Have fun and experiment with the app!
                """);
        System.out.println(DASHES);

        try {
            runConversation(scanner);
```

```
        } catch (Exception ex) {
            System.out.println("There was a problem running the scenario: " +
ex.getMessage());
        }

        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("Amazon Bedrock Converse API with Tool Use Feature
Scenario is complete.");
        System.out.println(DASHES);
    }

    /**
     * Starts the conversation with the user and handles the interaction with
Bedrock.
     */
    private static List<Message> runConversation(Scanner scanner) {
        List<Message> conversation = new ArrayList<>();

        // Get the first user input
        String userInput = getUserInput("Your weather info request:", scanner);
        System.out.println(userInput);

        while (userInput != null) {
            ContentBlock block = ContentBlock.builder()
                    .text(userInput)
                    .build();

            List<ContentBlock> blockList = new ArrayList<>();
            blockList.add(block);

            Message message = Message.builder()
                    .role(ConversationRole.USER)
                    .content(blockList)
                    .build();

            conversation.add(message);

            // Send the conversation to Amazon Bedrock.
            ConverseResponse bedrockResponse =
sendConversationToBedrock(conversation);
```

```
            // Recursively handle the model's response until the model has returned
its final response or the recursion counter has reached 0.
            processModelResponse(bedrockResponse, conversation, maxRecursions);

            // Repeat the loop until the user decides to exit the application.
            userInput = getUserInput("Your weather info request:", scanner);
        }
        printFooter();
        return conversation;
    }

    /**
     * Processes the response from the model and updates the conversation
accordingly.
     *
     * @param modelResponse the response from the model
     * @param conversation  the ongoing conversation
     * @param maxRecursion  the maximum number of recursions allowed
     */
    private static void processModelResponse(ConverseResponse modelResponse,
List<Message> conversation, int maxRecursion) {
        if (maxRecursion <= 0) {
            // Stop the process, the number of recursive calls could indicate an
infinite loop
            System.out.println("\tWarning: Maximum number of recursions reached.
Please try again.");
        }

        // Append the model's response to the ongoing conversation
        conversation.add(modelResponse.output().message());

        String modelResponseVal = modelResponse.stopReasonAsString();
        if (modelResponseVal.compareTo("tool_use") == 0) {
            // If the stop reason is "tool_use", forward everything to the tool use
handler
            handleToolUse(modelResponse.output(), conversation, maxRecursion - 1);
        }

        if (modelResponseVal.compareTo("end_turn") == 0) {
            // If the stop reason is "end_turn", print the model's response text,
and finish the process

PrintModelResponse(modelResponse.output().message().content().get(0).text());
            if (!interactive) {
```

```
                defaultPrompt = "x";
            }
        }
    }

    /**
     * Handles the use of a tool by the model in a conversation.
     *
     * @param modelResponse the response from the model, which may include a tool
use request
     * @param conversation  the current conversation, which will be updated with the
tool use results
     * @param maxRecursion  the maximum number of recursive calls allowed to handle
the model's response
     */
    private static void handleToolUse(ConverseOutput modelResponse, List<Message>
conversation, int maxRecursion) {
        List<ContentBlock> toolResults = new ArrayList<>();

        // The model's response can consist of multiple content blocks
        for (ContentBlock contentBlock : modelResponse.message().content()) {
            if (contentBlock.text() != null && !contentBlock.text().isEmpty()) {
                // If the content block contains text, print it to the console
                PrintModelResponse(contentBlock.text());
            }

            if (contentBlock.toolUse() != null) {
                ToolResponse toolResponse = invokeTool(contentBlock.toolUse());

                // Add the tool use ID and the tool's response to the list of
results
                List<ToolResultContentBlock> contentBlockList = new ArrayList<>();
                ToolResultContentBlock block = ToolResultContentBlock.builder()
                        .json(toolResponse.getContent())
                        .build();
                contentBlockList.add(block);

                ToolResultBlock toolResultBlock = ToolResultBlock.builder()
                        .toolUseId(toolResponse.getToolUseId())
                        .content(contentBlockList)
                        .build();

                ContentBlock contentBlock1 = ContentBlock.builder()
                        .toolResult(toolResultBlock)
```

```
                        .build();

                toolResults.add(contentBlock1);
            }
        }

        // Embed the tool results in a new user message
        Message message = Message.builder()
                .role(ConversationRole.USER)
                .content(toolResults)
                .build();

        // Append the new message to the ongoing conversation
        //conversation.add(message);
        conversation.add(message);

        // Send the conversation to Amazon Bedrock
        var response = sendConversationToBedrock(conversation);

        // Recursively handle the model's response until the model has returned its
final response or the recursion counter has reached 0
        processModelResponse(response, conversation, maxRecursion);
    }

    // Invokes the specified tool with the given payload and returns the tool's
response.
    // If the requested tool does not exist, an error message is returned.
    private static ToolResponse invokeTool(ToolUseBlock payload) {
        String toolName = payload.name();

        if (Objects.equals(toolName, "Weather_Tool")) {
            Map<String, Document> inputData = payload.input().asMap();
            printToolUse(toolName, inputData);

            // Invoke the weather tool with the input data provided
            Document weatherResponse =
weatherTool.fetchWeatherData(inputData.get("latitude").toString(),
inputData.get("longitude").toString());

            ToolResponse toolResponse = new ToolResponse();
            toolResponse.setContent(weatherResponse);
            toolResponse.setToolUseId(payload.toolUseId());
            return toolResponse;
        } else {
```

```
            String errorMessage = "The requested tool with name " + toolName + "
does not exist.";
            System.out.println(errorMessage);
            return null;
        }
    }

    public static void printToolUse(String toolName, Map<String, Document>
inputData) {
        System.out.println("Invoking tool: " + toolName + " with input: " +
inputData.get("latitude").toString() + ", " + inputData.get("longitude").toString()
+ "...");
    }

    private static void PrintModelResponse(String message) {
        System.out.println("\tThe model's response:\n");
        System.out.println(message);
        System.out.println("");
    }

    private static ConverseResponse sendConversationToBedrock(List<Message>
conversation) {
        System.out.println("Calling Bedrock...");

        try {
            return bedrockActions.sendConverseRequestAsync(modelId, systemPrompt,
conversation, weatherTool.getToolSpec());
        } catch (ModelNotReadyException ex) {
             System.err.println("Model is not ready. Please try again later: " +
ex.getMessage());
            throw ex;
        } catch (BedrockRuntimeException ex) {
            System.err.println("Bedrock service error: " + ex.getMessage());
            throw ex;
        } catch (RuntimeException ex) {
            System.err.println("Unexpected error occurred: " + ex.getMessage());
            throw ex;
        }
    }

    private static ConverseResponse sendConversationToBedrockwithSpec(List<Message>
conversation, ToolSpecification toolSpec) {
        System.out.println("Calling Bedrock...");
```

```
        // Send the conversation, system prompt, and tool configuration, and return
the response
        return bedrockActions.sendConverseRequestAsync(modelId, systemPrompt,
conversation, toolSpec);
    }

    public static String getUserInput(String prompt, Scanner scanner) {
        String userInput = defaultPrompt;
        if (interactive) {
            System.out.println("*".repeat(80));
            System.out.println(prompt + " (x to exit): \n\t");
            userInput = scanner.nextLine();
        }

        if (userInput == null || userInput.trim().isEmpty()) {
            return getUserInput("\tPlease enter your weather info request, e.g., the
name of a city", scanner);
        }

        if (userInput.equalsIgnoreCase("x")) {
            return null;
        }

        return userInput;
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }

    public static void printFooter() {
```

```
        System.out.println("""
                ================================================
                Thank you for checking out the Amazon Bedrock Tool Use demo. We hope
 you
                learned something new, or got some inspiration for your own apps
 today!

                For more Bedrock examples in different programming languages, have a
 look at:
                https://docs.aws.amazon.com/bedrock/latest/userguide/
service_code_examples.html
                ================================================
                """);
    }
}
```

演示使用的天气工具。该文件定义了工具规范，并实现了从 Open-Meteo API 中检索天气数据的逻辑。

```
public class WeatherTool {

    private static final Logger logger = LoggerFactory.getLogger(WeatherTool.class);
    private static java.net.http.HttpClient httpClient = null;

    /**
     * Returns the JSON Schema specification for the Weather tool. The tool
 specification
     * defines the input schema and describes the tool's functionality.
     * For more information, see https://json-schema.org/understanding-json-schema/
reference.
     *
     * @return The tool specification for the Weather tool.
     */
    public ToolSpecification getToolSpec() {
        Map<String, Document> latitudeMap = new HashMap<>();
        latitudeMap.put("type", Document.fromString("string"));
        latitudeMap.put("description", Document.fromString("Geographical WGS84
 latitude of the location."));

        // Create the nested "longitude" object
        Map<String, Document> longitudeMap = new HashMap<>();
        longitudeMap.put("type", Document.fromString("string"));
```

```
        longitudeMap.put("description", Document.fromString("Geographical WGS84
longitude of the location."));

        // Create the "properties" object
        Map<String, Document> propertiesMap = new HashMap<>();
        propertiesMap.put("latitude", Document.fromMap(latitudeMap));
        propertiesMap.put("longitude", Document.fromMap(longitudeMap));

        // Create the "required" array
        List<Document> requiredList = new ArrayList<>();
        requiredList.add(Document.fromString("latitude"));
        requiredList.add(Document.fromString("longitude"));

        // Create the root object
        Map<String, Document> rootMap = new HashMap<>();
        rootMap.put("type", Document.fromString("object"));
        rootMap.put("properties", Document.fromMap(propertiesMap));
        rootMap.put("required", Document.fromList(requiredList));

        // Now create the Document representing the JSON schema
        Document document = Document.fromMap(rootMap);

        ToolSpecification specification = ToolSpecification.builder()
            .name("Weather_Tool")
            .description("Get the current weather for a given location, based on its
WGS84 coordinates.")
            .inputSchema(ToolInputSchema.builder()
                .json(document)
                .build())
            .build();

        return specification;
    }

    /**
     * Fetches weather data for the given latitude and longitude.
     *
     * @param latitude  the latitude coordinate
     * @param longitude the longitude coordinate
     * @return a {@link CompletableFuture} containing the weather data as a JSON
string
     */
    public Document fetchWeatherData(String latitude, String longitude) {
        HttpClient httpClient = HttpClient.newHttpClient();
```

```
        // Ensure no extra double quotes
        latitude = latitude.replace("\"", "");
        longitude = longitude.replace("\"", "");

        String endpoint = "https://api.open-meteo.com/v1/forecast";
        String url = String.format("%s?latitude=%s&longitude=
%s&current_weather=True", endpoint, latitude, longitude);

        HttpRequest request = HttpRequest.newBuilder()
            .uri(URI.create(url))
            .build();

        try {
            HttpResponse<String> response = httpClient.send(request,
 HttpResponse.BodyHandlers.ofString());
            if (response.statusCode() == 200) {
                String weatherJson = response.body();
                System.out.println(weatherJson);
                ObjectMapper objectMapper = new ObjectMapper();
                Map<String, Object> rawMap = objectMapper.readValue(weatherJson, new
 TypeReference<Map<String, Object>>() {});
                Map<String, Document> documentMap = convertToDocumentMap(rawMap);


                Document weatherDocument = Document.fromMap(documentMap);
                System.out.println(weatherDocument);
                return weatherDocument;
            } else {
                throw new RuntimeException("Error fetching weather data: " +
 response.statusCode());
            }
        } catch (Exception e) {
            System.out.println("Error fetching weather data: " + e.getMessage());
            throw new RuntimeException("Error fetching weather data", e);
        }

    }

    private static Map<String, Document> convertToDocumentMap(Map<String, Object>
 inputMap) {
        Map<String, Document> result = new HashMap<>();
        for (Map.Entry<String, Object> entry : inputMap.entrySet()) {
            result.put(entry.getKey(), convertToDocument(entry.getValue()));
```

```
        }
        return result;
    }

    // Convert different types of Objects to Document
    private static Document convertToDocument(Object value) {
        if (value instanceof Map) {
            return Document.fromMap(convertToDocumentMap((Map<String, Object>)
 value));
        } else if (value instanceof Integer) {
            return Document.fromNumber(SdkNumber.fromInteger((Integer) value));
        } else if (value instanceof Double) {  //
            return Document.fromNumber(SdkNumber.fromDouble((Double) value));
        } else if (value instanceof Boolean) {
            return Document.fromBoolean((Boolean) value);
        } else if (value instanceof String) {
            return Document.fromString((String) value);
        }
        return Document.fromNull(); // Handle null values safely
    }
}
```

Converse API 操作和工具配置。

```
    /**
     * Sends an asynchronous converse request to the AI model.
     *
     * @param modelId      the unique identifier of the AI model to be used for the
 converse request
     * @param systemPrompt the system prompt to be included in the converse request
     * @param conversation a list of messages representing the conversation history
     * @param toolSpec     the specification of the tool to be used in the converse
 request
     * @return the converse response received from the AI model
     */
    public ConverseResponse sendConverseRequestAsync(String modelId, String
 systemPrompt, List<Message> conversation, ToolSpecification toolSpec) {
        List<Tool> toolList = new ArrayList<>();
        Tool tool = Tool.builder()
            .toolSpec(toolSpec)
            .build();
```

```
        toolList.add(tool);

        ToolConfiguration configuration = ToolConfiguration.builder()
            .tools(toolList)
            .build();

        SystemContentBlock block = SystemContentBlock.builder()
            .text(systemPrompt)
            .build();

        ConverseRequest request = ConverseRequest.builder()
            .modelId(modelId)
            .system(block)
            .messages(conversation)
            .toolConfig(configuration)
            .build();

        try {
            ConverseResponse response = getClient().converse(request).join();
            return response;

        } catch (ModelNotReadyException ex) {
            throw new RuntimeException("Model is not ready: " + ex.getMessage(),
ex);
        } catch (BedrockRuntimeException ex) {
            throw new RuntimeException("Failed to converse with Bedrock model: " +
ex.getMessage(), ex);
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 Converse。

Amazon Nova Canvas

InvokeModel

以下代码示例演示如何在 Amazon Bedrock 上调用 Amazon Nova Canvas 来生成图像。

适用于 Java 的 SDK 2.x

> **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。

使用 Amazon Nova Canvas 创建图像。

```
import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.InvokeModelResponse;

import java.security.SecureRandom;
import java.util.Base64;

import static com.example.bedrockruntime.libs.ImageTools.displayImage;

/**
 * This example demonstrates how to use Amazon Nova Canvas to generate images.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Configure the image generation parameters
 * - Send a request to generate an image
 * - Process the response and handle the generated image
 */
public class InvokeModel {

    public static byte[] invokeModel() {

        // Step 1: Create the Amazon Bedrock runtime client
        // The runtime client handles the communication with AI models on Amazon
 Bedrock
        BedrockRuntimeClient client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
```

```
                .build();

        // Step 2: Specify which model to use
        // For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        String modelId = "amazon.nova-canvas-v1:0";

        // Step 3: Configure the generation parameters and create the request
        // First, set the main parameters:
        // - prompt: Text description of the image to generate
        // - seed: Random number for reproducible generation (0 to 858,993,459)
        String prompt = "A stylized picture of a cute old steampunk robot";
        int seed = new SecureRandom().nextInt(858_993_460);

        // Then, create the request using a template with the following structure:
        // - taskType: TEXT_IMAGE (specifies text-to-image generation)
        // - textToImageParams: Contains the text prompt
        // - imageGenerationConfig: Contains optional generation settings (seed,
 quality, etc.)
        // For a list of available request parameters, see:
        // https://docs.aws.amazon.com/nova/latest/userguide/image-gen-req-resp-
structure.html
        String request = """
                {
                    "taskType": "TEXT_IMAGE",
                    "textToImageParams": {
                        "text": "{{prompt}}"
                    },
                    "imageGenerationConfig": {
                        "seed": {{seed}},
                        "quality": "standard"
                    }
                }"""
                .replace("{{prompt}}", prompt)
                .replace("{{seed}}", String.valueOf(seed));

        // Step 4: Send and process the request
        // - Send the request to the model using InvokeModelResponse
        // - Extract the Base64-encoded image from the JSON response
        // - Convert the encoded image to a byte array and return it
        try {
            InvokeModelResponse response = client.invokeModel(builder -> builder
                    .modelId(modelId)
```

```
                    .body(SdkBytes.fromUtf8String(request))
            );

            JSONObject responseBody = new
 JSONObject(response.body().asUtf8String());
            // Convert the Base64 string to byte array for better handling
            return Base64.getDecoder().decode(
                    new JSONPointer("/images/0").queryFrom(responseBody).toString()
            );

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s%n", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        System.out.println("Generating image. This may take a few seconds...");
        byte[] imageData = invokeModel();
        displayImage(imageData);
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考InvokeModel中的。

Amazon Titan 图像生成器

InvokeModel

以下代码示例演示如何在 Amazon Bedrock 上调用 Amazon Titan Image 来生成图像。

适用于 Java 的 SDK 2.x

> **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。

使用 Amazon Titan 图像生成器创建图像。

```
// Create an image with the Amazon Titan Image Generator.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

import java.math.BigInteger;
import java.security.SecureRandom;

import static com.example.bedrockruntime.libs.ImageTools.displayImage;

public class InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Titan Image G1.
        var modelId = "amazon.titan-image-generator-v1";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
 at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
titan-image.html
        var nativeRequestTemplate = """
                {
                    "taskType": "TEXT_IMAGE",
                    "textToImageParams": { "text": "{{prompt}}" },
                    "imageGenerationConfig": { "seed": {{seed}} }
                }""";
```

```
        // Define the prompt for the image generation.
        var prompt = "A stylized picture of a cute old steampunk robot";

        // Get a random 31-bit seed for the image generation (max. 2,147,483,647).
        var seed = new BigInteger(31, new SecureRandom());

        // Embed the prompt and seed in the model's native request payload.
        var nativeRequest = nativeRequestTemplate
                .replace("{{prompt}}", prompt)
                .replace("{{seed}}", seed.toString());

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                    .body(SdkBytes.fromUtf8String(nativeRequest))
                    .modelId(modelId)
            );

            // Decode the response body.
            var responseBody = new JSONObject(response.body().asUtf8String());

            // Retrieve the generated image data from the model's response.
            var base64ImageData = new JSONPointer("/
images/0").queryFrom(responseBody).toString();

            return base64ImageData;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        System.out.println("Generating image. This may take a few seconds...");

        String base64ImageData = invokeModel();

        displayImage(base64ImageData);
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考InvokeModel中的。

Amazon Titan 文本嵌入

InvokeModel

以下代码示例展示了如何：

- 开始创建您的第一个嵌入对象。
- 通过配置维度数量和标准化来创建嵌入对象（仅限 V2）。

适用于 Java 的 SDK 2.x

> **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。

使用 Titan 文本嵌入 V2 创建您的第一个嵌入对象。

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

通过配置维度数量和标准化来调用 Titan 文本嵌入 V2。

```
   /**
    * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考InvokeModel中的。

Anthropic Claude

Converse

以下代码示例演示如何使用 Bedrock 的 Converse API 向 Anthropic Claude 发送文本消息。

适用于 Java 的 SDK 2.x

> **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。

使用 Bedrock 的 Converse API 向 Anthropic Claude 发送文本消息。

```
// Use the Converse API to send a text message to Anthropic Claude.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
```

```
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.ConverseResponse;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

public class Converse {

    public static String converse() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // Create the input text and embed it in a message object with the user
 role.
        var inputText = "Describe the purpose of a 'hello world' program in one
 line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();


        try {
            // Send the message with a basic inference configuration.
            ConverseResponse response = client.converse(request -> request
                    .modelId(modelId)
                    .messages(message)
                    .inferenceConfig(config -> config
                            .maxTokens(512)
                            .temperature(0.5F)
                            .topP(0.9F)));

            // Retrieve the generated text from Bedrock's response object.
            var responseText =
 response.output().message().content().getFirst().text();
            System.out.println(responseText);
```

```
            return responseText;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        converse();
    }
}
```

将 Bedrock 的 Converse API 与异步 Java 客户端搭配使用向 Anthropic Claude 发送文本消息。

```
// Use the Converse API to send a text message to Anthropic Claude
// with the async Java client.

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.ContentBlock;
import software.amazon.awssdk.services.bedrockruntime.model.ConversationRole;
import software.amazon.awssdk.services.bedrockruntime.model.Message;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;

public class ConverseAsync {

    public static String converseAsync() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Claude 3 Haiku.
```

```
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // Create the input text and embed it in a message object with the user
role.
        var inputText = "Describe the purpose of a 'hello world' program in one
line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(inputText))
                .role(ConversationRole.USER)
                .build();

        // Send the message with a basic inference configuration.
        var request = client.converse(params -> params
                .modelId(modelId)
                .messages(message)
                .inferenceConfig(config -> config
                        .maxTokens(512)
                        .temperature(0.5F)
                        .topP(0.9F))
        );

        // Prepare a future object to handle the asynchronous response.
        CompletableFuture<String> future = new CompletableFuture<>();

        // Handle the response or error using the future object.
        request.whenComplete((response, error) -> {
            if (error == null) {
                // Extract the generated text from Bedrock's response object.
                String responseText =
response.output().message().content().getFirst().text();
                future.complete(responseText);
            } else {
                future.completeExceptionally(error);
            }
        });

        try {
            // Wait for the future object to complete and retrieve the generated
text.
            String responseText = future.get();
            System.out.println(responseText);

            return responseText;
```

```
        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        converseAsync();
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 Converse。

ConverseStream

以下代码示例演示如何使用 Bedrock 的 Converse API 向 Anthropic Claude 发送文本消息并实时处理响应流。

适用于 Java 的 SDK 2.x

> **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。

使用 Bedrock 的 Converse API 向 Anthropic Claude 发送文本消息并实时处理响应流。

```
// 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ConverseStream](#)中的。

InvokeModel

以下代码示例演示如何使用调用模型 API 向 Anthropic Claude 发送文本消息。

适用于 Java 的 SDK 2.x

> **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。

使用调用模型 API 发送文本消息。

```
// 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 的详细信息，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 API 向 Anthropic Claude 模型发送文本消息并打印响应流。

适用于 Java 的 SDK 2.x

> **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。

使用调用模型 API 发送文本消息并实时处理响应流。

```
// 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.InvokeModelWithResponseStreamResponseH

import java.util.Objects;
import java.util.concurrent.ExecutionException;

import static
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponseH

public class InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Claude 3 Haiku.
        var modelId = "anthropic.claude-3-haiku-20240307-v1:0";

        // The InvokeModelWithResponseStream API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
 at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
anthropic-claude-messages.html
        var nativeRequestTemplate = """
                {
                    "anthropic_version": "bedrock-2023-05-31",
                    "max_tokens": 512,
                    "temperature": 0.5,
                    "messages": [{
                        "role": "user",
                        "content": "{{prompt}}"
                    }]
                }""";
```

```
        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in the model's native request payload.
        String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

        // Create a request with the model ID and the model's native request
 payload.
        var request = InvokeModelWithResponseStreamRequest.builder()
                .body(SdkBytes.fromUtf8String(nativeRequest))
                .modelId(modelId)
                .build();

        // Prepare a buffer to accumulate the generated response text.
        var completeResponseTextBuffer = new StringBuilder();

        // Prepare a handler to extract, accumulate, and print the response text in
 real-time.
        var responseStreamHandler =
 InvokeModelWithResponseStreamResponseHandler.builder()
                .subscriber(Visitor.builder().onChunk(chunk -> {
                    var response = new JSONObject(chunk.bytes().asUtf8String());

                    // Extract and print the text from the content blocks.
                    if (Objects.equals(response.getString("type"),
 "content_block_delta")) {
                        var text = new JSONPointer("/delta/
text").queryFrom(response);
                        System.out.print(text);

                        // Append the text to the response text buffer.
                        completeResponseTextBuffer.append(text);
                    }
                }).build()).build();

        try {
            // Send the request and wait for the handler to process the response.
            client.invokeModelWithResponseStream(request,
 responseStreamHandler).get();

            // Return the complete response text.
            return completeResponseTextBuffer.toString();

        } catch (ExecutionException | InterruptedException e) {
```

```
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) throws ExecutionException,
 InterruptedException {
        invokeModelWithResponseStream();
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考InvokeModelWithResponseStream中的。

Reasoning

以下代码示例演示如何在 Amazon Bedrock 上使用 Anthropic Claude 3.7 Sonnet 的推理功能

适用于 Java 的 SDK 2.x

> **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。

将 Anthropic Claude 3.7 Sonnet 的推理功能与异步 Bedrock 运行时客户端结合使用。

```
import com.example.bedrockruntime.models.anthropicClaude.lib.ReasoningResponse;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

import java.util.concurrent.CompletableFuture;

/**
```

```
 * This example demonstrates how to use Anthropic Claude 3.7 Sonnet's reasoning
 capability
 * with an asynchronous Amazon Bedrock runtime client.
 * It shows how to:
 * - Set up the Amazon Bedrock async runtime client
 * - Create a message
 * - Configure reasoning parameters
 * - Send an asynchronous request with reasoning enabled
 * - Process both the reasoning output and final response
 */
public class ReasoningAsync {

    public static ReasoningResponse reasoningAsync() {

        // Create the Amazon Bedrock runtime client
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        var modelId = "us.anthropic.claude-3-7-sonnet-20250219-v1:0";

        // Create the message with the user's prompt
        var prompt = "Describe the purpose of a 'hello world' program in one line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(prompt))
                .role(ConversationRole.USER)
                .build();

        // Configure reasoning parameters with a 2000 token budget
        Document reasoningConfig = Document.mapBuilder()
                .putDocument("thinking", Document.mapBuilder()
                        .putString("type", "enabled")
                        .putNumber("budget_tokens", 2000)
                        .build())
                .build();

        try {
            // Send message and reasoning configuration to the model
            CompletableFuture<ConverseResponse> asyncResponse =
 client.converse(request -> request
```

```
                    .additionalModelRequestFields(reasoningConfig)
                    .messages(message)
                    .modelId(modelId)
            );

            // Process the response asynchronously
            return asyncResponse.thenApply(response -> {

                        var content = response.output().message().content();
                        ReasoningContentBlock reasoning = null;
                        String text = null;

                        // Process each content block to find reasoning and response
 text
                        for (ContentBlock block : content) {
                            if (block.reasoningContent() != null) {
                                reasoning = block.reasoningContent();
                            } else if (block.text() != null) {
                                text = block.text();
                            }
                        }

                        return new ReasoningResponse(reasoning, text);
                    }
            ).get();

        } catch (Exception e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        // Execute the example and display reasoning and final response
        ReasoningResponse response = reasoningAsync();
        System.out.println("\n<thinking>");
        System.out.println(response.reasoning().reasoningText());
        System.out.println("</thinking>\n");
        System.out.println(response.text());
    }
}
```

将 Anthropic Claude 3.7 Sonnet 的推理功能与同步 Bedrock 运行时客户端结合使用。

```
import com.example.bedrockruntime.models.anthropicClaude.lib.ReasoningResponse;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

/**
 * This example demonstrates how to use Anthropic Claude 3.7 Sonnet's reasoning
 capability
 * with the synchronous Amazon Bedrock runtime client.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure reasoning parameters
 * - Send a request with reasoning enabled
 * - Process both the reasoning output and final response
 */
public class Reasoning {

    public static ReasoningResponse reasoning() {

        // Create the Amazon Bedrock runtime client
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        var modelId = "us.anthropic.claude-3-7-sonnet-20250219-v1:0";

        // Create the message with the user's prompt
        var prompt = "Describe the purpose of a 'hello world' program in one line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(prompt))
                .role(ConversationRole.USER)
                .build();
```

```
        // Configure reasoning parameters with a 2000 token budget
        Document reasoningConfig = Document.mapBuilder()
                .putDocument("thinking", Document.mapBuilder()
                        .putString("type", "enabled")
                        .putNumber("budget_tokens", 2000)
                        .build())
                .build();

        try {
            // Send message and reasoning configuration to the model
            ConverseResponse bedrockResponse = client.converse(request -> request
                    .additionalModelRequestFields(reasoningConfig)
                    .messages(message)
                    .modelId(modelId)
            );


            // Extract both reasoning and final response
            var content = bedrockResponse.output().message().content();
            ReasoningContentBlock reasoning = null;
            String text = null;

            // Process each content block to find reasoning and response text
            for (ContentBlock block : content) {
                if (block.reasoningContent() != null) {
                    reasoning = block.reasoningContent();
                } else if (block.text() != null) {
                    text = block.text();
                }
            }

            return new ReasoningResponse(reasoning, text);

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        // Execute the example and display reasoning and final response
        ReasoningResponse response = reasoning();
        System.out.println("\n<thinking>");
```

```
        System.out.println(response.reasoning().reasoningText());
        System.out.println("</thinking>\n");
        System.out.println(response.text());
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 Converse。

使用流式响应进行推理

以下代码示例演示如何在 Amazon Bedrock 上使用 Anthropic Claude 3.7 Sonnet 的推理能力

适用于 Java 的 SDK 2.x

> **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。

使用 Anthropic Claude 3.7 Sonnet 的推理功能生成流式文字响应。

```
import com.example.bedrockruntime.models.anthropicClaude.lib.ReasoningResponse;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeAsyncClient;
import software.amazon.awssdk.services.bedrockruntime.model.*;

import java.util.concurrent.ExecutionException;
import java.util.concurrent.atomic.AtomicReference;

/**
 * This example demonstrates how to use Anthropic Claude 3.7 Sonnet's reasoning
 * capability to generate streaming text responses.
 * It shows how to:
 * - Set up the Amazon Bedrock runtime client
 * - Create a message
 * - Configure a streaming request
```

```
 * - Set up a stream handler to process the response chunks
 * - Process the streaming response
 */
public class ReasoningStream {

    public static ReasoningResponse reasoningStream() {

        // Create the Amazon Bedrock runtime client
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Specify the model ID. For the latest available models, see:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/models-
supported.html
        var modelId = "us.anthropic.claude-3-7-sonnet-20250219-v1:0";

        // Create the message with the user's prompt
        var prompt = "Describe the purpose of a 'hello world' program in one line.";
        var message = Message.builder()
                .content(ContentBlock.fromText(prompt))
                .role(ConversationRole.USER)
                .build();

        // Configure reasoning parameters with a 2000 token budget
        Document reasoningConfig = Document.mapBuilder()
                .putDocument("thinking", Document.mapBuilder()
                        .putString("type", "enabled")
                        .putNumber("budget_tokens", 2000)
                        .build())
                .build();

        // Configure the request with the message, model ID, and reasoning config
        ConverseStreamRequest request = ConverseStreamRequest.builder()
                .additionalModelRequestFields(reasoningConfig)
                .messages(message)
                .modelId(modelId)
                .build();

        StringBuilder reasoning = new StringBuilder();
        StringBuilder text = new StringBuilder();
        AtomicReference<ReasoningResponse> finalresponse = new AtomicReference<>();
```

```
        // Set up the stream handler to processes chunks of the response as they
arrive
        var streamHandler = ConverseStreamResponseHandler.builder()
                .subscriber(ConverseStreamResponseHandler.Visitor.builder()
                        .onContentBlockDelta(chunk -> {
                            ContentBlockDelta delta = chunk.delta();
                            if (delta.reasoningContent() != null) {
                                if (reasoning.isEmpty()) {
                                    System.out.println("\n<thinking>");
                                }
                                if (delta.reasoningContent().text() != null) {

System.out.print(delta.reasoningContent().text());

reasoning.append(delta.reasoningContent().text());
                                }
                            } else if (delta.text() != null) {
                                if (text.isEmpty()) {
                                    System.out.println("\n</thinking>\n");
                                }
                                System.out.print(delta.text());
                                text.append(delta.text());
                            }
                            System.out.flush();  // Ensure immediate output of each
chunk
                        }).build())
                .onComplete(() -> finalresponse.set(new ReasoningResponse(
                        ReasoningContentBlock.fromReasoningText(t ->
t.text(reasoning.toString())),
                        text.toString()
                )))
                .onError(err -> System.err.printf("Can't invoke '%s': %s", modelId,
err.getMessage()))
                .build();

        // Step 6: Send the streaming request and process the response
        // - Send the request to the model
        // - Attach the handler to process response chunks as they arrive
        // - Handle any errors during streaming
        try {
            client.converseStream(request, streamHandler).get();
            return finalresponse.get();

        } catch (ExecutionException | InterruptedException e) {
```

```
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
            throw new RuntimeException(e);
        } catch (Exception e) {
            System.err.printf("Can't invoke '%s': %s", modelId, e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        reasoningStream();
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 Converse。

Cohere Command

Converse

以下代码示例演示如何使用 Bedrock 的 Converse API 向 Cohere Command 发送文本消息。

适用于 Java 的 SDK 2.x

> **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。

使用 Bedrock 的 Converse API 向 Cohere Command 发送文本消息。

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

将 Bedrock 的 Converse API 与异步 Java 客户端搭配使用向 Cohere Command 发送文本消息。

```
// 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 Reference》中的 Converse。

ConverseStream

以下代码示例演示如何使用 Bedrock 的 Converse API 向 Cohere Command 发送文本消息并实时处理响应流。

适用于 Java 的 SDK 2.x

> **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。

使用 Bedrock 的 Converse API 向 Cohere Command 发送文本消息并实时处理响应流。

```
// 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ConverseStream](#)中的。

InvokeModel: 命令 R 和 R+

以下代码示例演示如何使用调用模型 API 向 Cohere Command R 和 R+ 发送文本消息。

适用于 Java 的 SDK 2.x

> **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。

使用调用模型 API 发送文本消息。

```
// 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 的详细信息，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通过将调用模型 API 与响应流结合使用，来向 Cohere Command 发送文本消息。

适用于 Java 的 SDK 2.x

> **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。

使用调用模型 API 发送文本消息并实时处理响应流。

```
// 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.InvokeModelWithResponseStreamResponseH

import java.util.concurrent.ExecutionException;
```

```
import static
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponseH

public class Command_R_InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Command R.
        var modelId = "cohere.command-r-v1:0";

        // The InvokeModelWithResponseStream API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
 at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
cohere-command-r-plus.html
        var nativeRequestTemplate = "{ \"message\": \"{{prompt}}\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in the model's native request payload.
        String nativeRequest = nativeRequestTemplate.replace("{{prompt}}", prompt);

        // Create a request with the model ID and the model's native request
 payload.
        var request = InvokeModelWithResponseStreamRequest.builder()
                .body(SdkBytes.fromUtf8String(nativeRequest))
                .modelId(modelId)
                .build();

        // Prepare a buffer to accumulate the generated response text.
        var completeResponseTextBuffer = new StringBuilder();

        // Prepare a handler to extract, accumulate, and print the response text in
 real-time.
```

```
        var responseStreamHandler =
 InvokeModelWithResponseStreamResponseHandler.builder()
                .subscriber(Visitor.builder().onChunk(chunk -> {
                    // Extract and print the text from the model's native response.
                    var response = new JSONObject(chunk.bytes().asUtf8String());
                    var text = new JSONPointer("/text").queryFrom(response);
                    System.out.print(text);

                    // Append the text to the response text buffer.
                    completeResponseTextBuffer.append(text);
                }).build()).build();

        try {
            // Send the request and wait for the handler to process the response.
            client.invokeModelWithResponseStream(request,
 responseStreamHandler).get();

            // Return the complete response text.
            return completeResponseTextBuffer.toString();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) throws ExecutionException,
 InterruptedException {
        invokeModelWithResponseStream();
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考InvokeModel中的。

Meta Llama

Converse

以下代码示例演示如何使用 Bedrock 的 Converse API 向 Meta Llama 发送文本消息。

适用于 Java 的 SDK 2.x

> **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。

使用 Bedrock 的 Converse API 向 Meta Llama 发送文本消息。

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

将 Bedrock 的 Converse API 与异步 Java 客户端搭配使用向 Meta Llama 发送文本消息。

```
// 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 Reference》中的 Converse。

ConverseStream

以下代码示例演示如何使用 Bedrock 的 Converse API 向 Meta Llama 发送文本消息并实时处理响应流。

适用于 Java 的 SDK 2.x

> **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。

使用 Bedrock 的 Converse API 向 Meta Llama 发送文本消息并实时处理响应流。

```
// 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ConverseStream中的。

InvokeModel

以下代码示例演示如何使用调用模型 API 向 Meta Llama 发送文本消息。

适用于 Java 的 SDK 2.x

> **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。

使用调用模型 API 发送文本消息。

```
// Use the native inference API to send a text message to Meta Llama 3.

import org.json.JSONObject;
import org.json.JSONPointer;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.bedrockruntime.BedrockRuntimeClient;

public class Llama3_InvokeModel {

    public static String invokeModel() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_WEST_2)
                .build();

        // Set the model ID, e.g., Llama 3 70b Instruct.
        var modelId = "meta.llama3-70b-instruct-v1:0";

        // The InvokeModel API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
 at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
meta.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

        // Define the prompt for the model.
```

```
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in Llama 3's instruction format.
        var instruction = (
                "<|begin_of_text|><|start_header_id|>user<|end_header_id|>\\n" +
                "{{prompt}} <|eot_id|>\\n" +
                "<|start_header_id|>assistant<|end_header_id|>\\n"
        ).replace("{{prompt}}", prompt);

        // Embed the instruction in the the native request payload.
        var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
 instruction);

        try {
            // Encode and send the request to the Bedrock Runtime.
            var response = client.invokeModel(request -> request
                    .body(SdkBytes.fromUtf8String(nativeRequest))
                    .modelId(modelId)
            );

            // Decode the response body.
            var responseBody = new JSONObject(response.body().asUtf8String());

            // Retrieve the generated text from the model's response.
            var text = new JSONPointer("/
generation").queryFrom(responseBody).toString();
            System.out.println(text);

            return text;

        } catch (SdkClientException e) {
            System.err.printf("ERROR: Can't invoke '%s'. Reason: %s", modelId,
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) {
        invokeModel();
    }
}
```

- 有关 API 的详细信息，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 API 向 Meta Llama 发送文本消息并打印响应流。

适用于 Java 的 SDK 2.x

> **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。

使用调用模型 API 发送文本消息并实时处理响应流。

```
// 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.InvokeModelWithResponseStreamResponseH

import java.util.concurrent.ExecutionException;

import static
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponseH

public class Llama3_InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_WEST_2)
```

```
                .build();

        // Set the model ID, e.g., Llama 3 70b Instruct.
        var modelId = "meta.llama3-70b-instruct-v1:0";

        // The InvokeModelWithResponseStream API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
 at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
meta.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

        // Define the prompt for the model.
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in Llama 3's instruction format.
        var instruction = (
                "<|begin_of_text|><|start_header_id|>user<|end_header_id|>\\n" +
                "{{prompt}} <|eot_id|>\\n" +
                "<|start_header_id|>assistant<|end_header_id|>\\n"
        ).replace("{{prompt}}", prompt);

        // Embed the instruction in the the native request payload.
        var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
 instruction);

        // Create a request with the model ID and the model's native request
 payload.
        var request = InvokeModelWithResponseStreamRequest.builder()
                .body(SdkBytes.fromUtf8String(nativeRequest))
                .modelId(modelId)
                .build();

        // Prepare a buffer to accumulate the generated response text.
        var completeResponseTextBuffer = new StringBuilder();

        // Prepare a handler to extract, accumulate, and print the response text in
 real-time.
        var responseStreamHandler =
 InvokeModelWithResponseStreamResponseHandler.builder()
                .subscriber(Visitor.builder().onChunk(chunk -> {
                    // Extract and print the text from the model's native response.
                    var response = new JSONObject(chunk.bytes().asUtf8String());
                    var text = new JSONPointer("/generation").queryFrom(response);
```

```
                    System.out.print(text);

                    // Append the text to the response text buffer.
                    completeResponseTextBuffer.append(text);
                }).build()).build();

        try {
            // Send the request and wait for the handler to process the response.
            client.invokeModelWithResponseStream(request,
 responseStreamHandler).get();

            // Return the complete response text.
            return completeResponseTextBuffer.toString();

        } catch (ExecutionException | InterruptedException e) {
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) throws ExecutionException,
 InterruptedException {
        invokeModelWithResponseStream();
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[InvokeModelWithResponseStream](#)中的。

Mistral AI

Converse

以下代码示例演示如何使用 Bedrock 的 Converse API 向 Mistral 发送文本消息。

适用于 Java 的 SDK 2.x

> **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。

使用 Bedrock 的 Converse API 向 Mistral 发送文本消息。

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

将 Bedrock 的 Converse API 与异步 Java 客户端搭配使用向 Mistral 发送文本消息。

```
// 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 Reference》中的 Converse。

ConverseStream

以下代码示例演示如何使用 Bedrock 的 Converse API 向 Mistral 发送文本消息并实时处理响应流。

适用于 Java 的 SDK 2.x

> **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。

使用 Bedrock 的 Converse API 向 Mistral 发送文本消息并实时处理响应流。

```
// 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ConverseStream中的。

InvokeModel

以下代码示例演示如何使用调用模型 API 向 Mistral 模型发送文本消息。

适用于 Java 的 SDK 2.x

> **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。

使用调用模型 API 发送文本消息。

```
// 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 的详细信息，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 API 向 Mistral AI 模型发送文本消息并打印响应流。

适用于 Java 的 SDK 2.x

> **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。

使用调用模型 API 发送文本消息并实时处理响应流。

```
// 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.InvokeModelWithResponseStreamResponseH

import java.util.concurrent.ExecutionException;

import static
 software.amazon.awssdk.services.bedrockruntime.model.InvokeModelWithResponseStreamResponseH

public class InvokeModelWithResponseStream {

    public static String invokeModelWithResponseStream() {

        // Create a Bedrock Runtime client in the AWS Region you want to use.
        // Replace the DefaultCredentialsProvider with your preferred credentials
 provider.
        var client = BedrockRuntimeAsyncClient.builder()
                .credentialsProvider(DefaultCredentialsProvider.create())
                .region(Region.US_EAST_1)
                .build();

        // Set the model ID, e.g., Mistral Large.
        var modelId = "mistral.mistral-large-2402-v1:0";

        // The InvokeModelWithResponseStream API uses the model's native payload.
        // Learn more about the available inference parameters and response fields
 at:
        // https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-
mistral-text-completion.html
        var nativeRequestTemplate = "{ \"prompt\": \"{{instruction}}\" }";

        // Define the prompt for the model.
```

```
        var prompt = "Describe the purpose of a 'hello world' program in one line.";

        // Embed the prompt in Mistral's instruction format.
        var instruction = "<s>[INST] {{prompt}} [/INST]\\n".replace("{{prompt}}",
 prompt);

        // Embed the instruction in the the native request payload.
        var nativeRequest = nativeRequestTemplate.replace("{{instruction}}",
 instruction);

        // Create a request with the model ID and the model's native request
 payload.
        var request = InvokeModelWithResponseStreamRequest.builder()
                .body(SdkBytes.fromUtf8String(nativeRequest))
                .modelId(modelId)
                .build();

        // Prepare a buffer to accumulate the generated response text.
        var completeResponseTextBuffer = new StringBuilder();

        // Prepare a handler to extract, accumulate, and print the response text in
 real-time.
        var responseStreamHandler =
 InvokeModelWithResponseStreamResponseHandler.builder()
                .subscriber(Visitor.builder().onChunk(chunk -> {
                    // Extract and print the text from the model's native response.
                    var response = new JSONObject(chunk.bytes().asUtf8String());
                    var text = new JSONPointer("/outputs/0/
text").queryFrom(response);
                    System.out.print(text);

                    // Append the text to the response text buffer.
                    completeResponseTextBuffer.append(text);
                }).build()).build();

        try {
            // Send the request and wait for the handler to process the response.
            client.invokeModelWithResponseStream(request,
 responseStreamHandler).get();

            // Return the complete response text.
            return completeResponseTextBuffer.toString();

        } catch (ExecutionException | InterruptedException e) {
```

```
            System.err.printf("Can't invoke '%s': %s", modelId,
 e.getCause().getMessage());
            throw new RuntimeException(e);
        }
    }

    public static void main(String[] args) throws ExecutionException,
 InterruptedException {
        invokeModelWithResponseStream();
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[InvokeModelWithResponseStream](#)中的。

Stable Diffusion

InvokeModel

以下代码示例演示如何在 Amazon Bedrock 上调用 Stability.ai Stable Diffusion XL 来生成图像。

适用于 Java 的 SDK 2.x

> **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。

使用 Stable Diffusion 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考InvokeModel中的。

CloudFront 使用适用于 Java 的 SDK 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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从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`

以下代码示例演示了如何使用 `CreateDistribution`。

适用于 Java 的 SDK 2.x

> **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。

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))
                                                                    .allowedMethods(b4 -
> b4
  .quantity(2)
  .items(Method.HEAD, Method.GET)
  .cachedMethods(b5 -> b5
            .quantity(2)
            .items(Method.HEAD,
                                Method.GET))))
```

```
                                                .cacheBehaviors(b -> b
                                                        .quantity(1)
                                                        .items(b2 -> b2

.pathPattern("/index.html")

.viewerProtocolPolicy(

           ViewerProtocolPolicy.ALLOW_ALL)

.targetOriginId(originId)

.trustedKeyGroups(b3 -> b3

           .quantity(1)

           .items(keyGroupId)

           .enabled(true))

.minTTL(200L)

.forwardedValues(b4 -> b4

           .cookies(cp -> cp

                           .forward(ItemSelection.NONE))

           .queryString(true))

.allowedMethods(b5 -> b5.quantity(2)

           .items(Method.HEAD,

                          Method.GET)

           .cachedMethods(b6 -> b6

                           .quantity(2)

                           .items(Method.HEAD,

                                          Method.GET)))))
                                              .enabled(true)
```

```
                                                .comment("Distribution built with
 java")

 .callerReference(Instant.now().toString())));

                final Distribution distribution = createDistResponse.distribution();
                logger.info("Distribution created. DomainName: [{}]  Id: [{}]",
 distribution.domainName(),
                                    distribution.id());
                logger.info("Waiting for distribution to be deployed ...");
                try (CloudFrontWaiter cfWaiter =
 CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
                        ResponseOrException<GetDistributionResponse>
 responseOrException = cfWaiter
                                        .waitUntilDistributionDeployed(builder ->
 builder.id(distribution.id()))
                                        .matched();
                        responseOrException.response()
                                        .orElseThrow(() -> new
 RuntimeException("Distribution not created"));
                        logger.info("Distribution deployed. DomainName: [{}]  Id:
 [{}]", distribution.domainName(),
                                        distribution.id());
                }
                return distribution;
        }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateDistribution中的。

`CreateFunction`

以下代码示例演示了如何使用 `CreateFunction`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateFunction中的。

CreateKeyGroup

以下代码示例演示了如何使用 CreateKeyGroup。

适用于 Java 的 SDK 2.x

> **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。

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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateKeyGroup中的。

`CreatePublicKey`

以下代码示例演示了如何使用 `CreatePublicKey`。

适用于 Java 的 SDK 2.x

> **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。

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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreatePublicKey中的。

`DeleteDistribution`

以下代码示例演示了如何使用 `DeleteDistribution`。

适用于 Java 的 SDK 2.x

> **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。

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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteDistribution中的。

`UpdateDistribution`

以下代码示例演示了如何使用 `UpdateDistribution`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考UpdateDistribution中的。

场景

创建多租户分配和分配租户

以下代码示例演示了如何使用各种配置创建多租户分配和分配租户。

适用于 Java 的 SDK 2.x

> **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。

以下示例演示了如何使用参数和通配符证书创建多租户分配。

```
import software.amazon.awssdk.core.internal.waiters.ResponseOrException;
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import software.amazon.awssdk.services.cloudfront.model.ConnectionMode;
import software.amazon.awssdk.services.cloudfront.model.CreateDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.Distribution;
import software.amazon.awssdk.services.cloudfront.model.GetDistributionResponse;
import software.amazon.awssdk.services.cloudfront.model.HttpVersion;
import software.amazon.awssdk.services.cloudfront.model.Method;
import software.amazon.awssdk.services.cloudfront.model.SSLSupportMethod;
import software.amazon.awssdk.services.cloudfront.model.ViewerProtocolPolicy;
import software.amazon.awssdk.services.cloudfront.waiters.CloudFrontWaiter;
import software.amazon.awssdk.services.s3.S3Client;

import java.time.Instant;

public class CreateMultiTenantDistribution {
    public static Distribution
 CreateMultiTenantDistributionWithCert(CloudFrontClient cloudFrontClient,
                                                                    S3Client
 s3Client,
                                                                    final String
 bucketName,
                                                                    final String
 certificateArn) {
        // fetch the origin info if necessary
```

```
        final String region = s3Client.headBucket(b ->
 b.bucket(bucketName)).sdkHttpResponse().headers()
                .get("x-amz-bucket-region").get(0);
        final String originDomain = bucketName + ".s3." + region + ".amazonaws.com";
        String originId = originDomain; // Use the originDomain value for the
 originId.

        CreateDistributionResponse createDistResponse =
 cloudFrontClient.createDistribution(builder -> builder
                .distributionConfig(b1 -> b1
                        .httpVersion(HttpVersion.HTTP2)
                        .enabled(true)
                        .comment("Template Distribution with cert built with java")
                        .connectionMode(ConnectionMode.TENANT_ONLY)
                        .callerReference(Instant.now().toString())
                        .viewerCertificate(certBuilder -> certBuilder
                                .acmCertificateArn(certificateArn)
                                .sslSupportMethod(SSLSupportMethod.SNI_ONLY))
                        .origins(b2 -> b2
                                .quantity(1)
                                .items(b3 -> b3
                                        .domainName(originDomain)
                                        .id(originId)
                                        .originPath("/{{tenantName}}")
                                        .s3OriginConfig(builder4 -> builder4
                                                .originAccessIdentity(
                                                        ""))))
                        .tenantConfig(b5 -> b5
                                .parameterDefinitions(b6 -> b6
                                        .name("tenantName")
                                        .definition(b7 -> b7
                                                .stringSchema(b8 -> b8
                                                        .comment("tenantName value")
                                                        .defaultValue("root")
                                                        .required(false)))))
                        .defaultCacheBehavior(b2 -> b2

 .viewerProtocolPolicy(ViewerProtocolPolicy.ALLOW_ALL)
                                .targetOriginId(originId)
                                .cachePolicyId("658327ea-f89d-4fab-
a63d-7e88639e58f6") // CachingOptimized Policy
                                .allowedMethods(b4 -> b4
                                        .quantity(2)
                                        .items(Method.HEAD, Method.GET)))
```

```
                ));

        final Distribution distribution = createDistResponse.distribution();
        try (CloudFrontWaiter cfWaiter =
CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
            ResponseOrException<GetDistributionResponse> responseOrException =
cfWaiter
                    .waitUntilDistributionDeployed(builder ->
builder.id(distribution.id()))
                    .matched();
            responseOrException.response()
                    .orElseThrow(() -> new RuntimeException("Distribution not
created"));
        }
        return distribution;
    }

    public static Distribution CreateMultiTenantDistributionNoCert(CloudFrontClient
cloudFrontClient,
                                                                   S3Client s3Client,
                                                                   final String
bucketName) {
        // fetch the origin info if necessary
        final String region = s3Client.headBucket(b ->
b.bucket(bucketName)).sdkHttpResponse().headers()
                .get("x-amz-bucket-region").get(0);
        final String originDomain = bucketName + ".s3." + region + ".amazonaws.com";
        String originId = originDomain; // Use the originDomain value for the
originId.

        CreateDistributionResponse createDistResponse =
cloudFrontClient.createDistribution(builder -> builder
                .distributionConfig(b1 -> b1
                        .httpVersion(HttpVersion.HTTP2)
                        .enabled(true)
                        .comment("Template Distribution with cert built with java")
                        .connectionMode(ConnectionMode.TENANT_ONLY)
                        .callerReference(Instant.now().toString())
                        .origins(b2 -> b2
                                .quantity(1)
                                .items(b3 -> b3
                                        .domainName(originDomain)
                                        .id(originId)
                                        .originPath("/{{tenantName}}")
```

```
                                        .s3OriginConfig(builder4 -> builder4
                                                .originAccessIdentity(
                                                        "")))))
                        .tenantConfig(b5 -> b5
                                .parameterDefinitions(b6 -> b6
                                        .name("tenantName")
                                        .definition(b7 -> b7
                                                .stringSchema(b8 -> b8
                                                        .comment("tenantName value")
                                                        .defaultValue("root")
                                                        .required(false)))))
                        .defaultCacheBehavior(b2 -> b2

 .viewerProtocolPolicy(ViewerProtocolPolicy.ALLOW_ALL)
                                .targetOriginId(originId)
                                .cachePolicyId("658327ea-f89d-4fab-
a63d-7e88639e58f6") // CachingOptimized Policy
                                .allowedMethods(b4 -> b4
                                        .quantity(2)
                                        .items(Method.HEAD, Method.GET)))
                ));

        final Distribution distribution = createDistResponse.distribution();
        try (CloudFrontWaiter cfWaiter =
 CloudFrontWaiter.builder().client(cloudFrontClient).build()) {
            ResponseOrException<GetDistributionResponse> responseOrException =
 cfWaiter
                    .waitUntilDistributionDeployed(builder ->
 builder.id(distribution.id()))
                    .matched();
            responseOrException.response()
                    .orElseThrow(() -> new RuntimeException("Distribution not
 created"));
        }
        return distribution;
    }
}
```

以下示例演示了如何创建与该模板关联的分配租户，包括利用我们在上面声明的参数。请注意，我们不需要在此处添加证书信息，因为父模板涵盖了我们的域。

```
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.CreateConnectionGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionTenantResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionTenant;
import software.amazon.awssdk.services.cloudfront.model.GetConnectionGroupResponse;
import software.amazon.awssdk.services.cloudfront.model.ValidationTokenHost;
import software.amazon.awssdk.services.route53.Route53Client;
import software.amazon.awssdk.services.route53.model.RRType;

import java.time.Instant;

public class CreateDistributionTenant {

    public static DistributionTenant createDistributionTenantNoCert(CloudFrontClient
 cloudFrontClient,
                                                                     Route53Client
 route53Client,
                                                                     String
 distributionId,
                                                                     String domain,
                                                                     String
 hostedZoneId) {
        CreateDistributionTenantResponse createResponse =
 cloudFrontClient.createDistributionTenant(builder -> builder
                .distributionId(distributionId)
                .domains(b1 -> b1
                        .domain(domain))
                .parameters(b2 -> b2
                        .name("tenantName")
                        .value("myTenant"))
                .enabled(false)
                .name("no-cert-tenant")
        );

        final DistributionTenant distributionTenant =
 createResponse.distributionTenant();

        // Then update the Route53 hosted zone to point your domain at the
 distribution tenant
        // We fetch the RoutingEndpoint to point to via the default connection group
 that was created for your tenant
```

```
        final GetConnectionGroupResponse fetchedConnectionGroup =
 cloudFrontClient.getConnectionGroup(builder -> builder
                .identifier(distributionTenant.connectionGroupId()));

        route53Client.changeResourceRecordSets(builder -> builder
                .hostedZoneId(hostedZoneId)
                .changeBatch(b1 -> b1
                        .comment("ChangeBatch comment")
                        .changes(b2 -> b2
                                .resourceRecordSet(b3 -> b3
                                        .name(domain)
                                        .type("CNAME")
                                        .ttl(300L)
                                        .resourceRecords(b4 -> b4

 .value(fetchedConnectionGroup.connectionGroup().routingEndpoint())))
                                .action("CREATE"))
                ));
        return distributionTenant;
    }

}
```

如果父模板中省略了查看器证书，则需要改为添加与之关联的租户的证书信息。以下示例演示了如何通过涵盖租户所需域的 ACM 证书 arn 来执行此操作。

```
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.CreateConnectionGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionTenantResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionTenant;
import software.amazon.awssdk.services.cloudfront.model.GetConnectionGroupResponse;
import software.amazon.awssdk.services.cloudfront.model.ValidationTokenHost;
import software.amazon.awssdk.services.route53.Route53Client;
import software.amazon.awssdk.services.route53.model.RRType;

import java.time.Instant;

public class CreateDistributionTenant {
```

```
    public static DistributionTenant
 createDistributionTenantWithCert(CloudFrontClient cloudFrontClient,
                                                                    Route53Client
 route53Client,
                                                                    String
 distributionId,
                                                                    String domain,
                                                                    String
 hostedZoneId,
                                                                    String
 certificateArn) {
        CreateDistributionTenantResponse createResponse =
 cloudFrontClient.createDistributionTenant(builder -> builder
                .distributionId(distributionId)
                .domains(b1 -> b1
                        .domain(domain))
                .enabled(false)
                .name("tenant-with-cert")
                .parameters(b2 -> b2
                        .name("tenantName")
                        .value("myTenant"))
                .customizations(b3 -> b3
                        .certificate(b4 -> b4
                                .arn(certificateArn))) // NOTE: Cert must be in Us-
East-1 and cover the domain provided in this request

        );

        final DistributionTenant distributionTenant =
 createResponse.distributionTenant();

        // Then update the Route53 hosted zone to point your domain at the
 distribution tenant
        // We fetch the RoutingEndpoint to point to via the default connection group
 that was created for your tenant
        final GetConnectionGroupResponse fetchedConnectionGroup =
 cloudFrontClient.getConnectionGroup(builder -> builder
                .identifier(distributionTenant.connectionGroupId()));

        route53Client.changeResourceRecordSets(builder -> builder
                .hostedZoneId(hostedZoneId)
                .changeBatch(b1 -> b1
                        .comment("ChangeBatch comment")
                        .changes(b2 -> b2
```

```
                                .resourceRecordSet(b3 -> b3
                                        .name(domain)
                                        .type("CNAME")
                                        .ttl(300L)
                                        .resourceRecords(b4 -> b4

 .value(fetchedConnectionGroup.connectionGroup().routingEndpoint())))
                                .action("CREATE"))
                ));
        return distributionTenant;
    }

}
```

以下示例演示如何使用托管的 CloudFront托管证书请求执行此操作。如果还没有流量流向您的域，则这是理想之选。在本例中，我们创建 a ConnectionGroup 来生成 RoutingEndpoint。然后我们用它 RoutingEndpoint 来创建 DNS 记录，用于验证域名所有权并指向 CloudFront。 CloudFront 然后将自动提供令牌来验证域名所有权并创建托管证书。

```
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.CreateConnectionGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionTenantResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionTenant;
import software.amazon.awssdk.services.cloudfront.model.GetConnectionGroupResponse;
import software.amazon.awssdk.services.cloudfront.model.ValidationTokenHost;
import software.amazon.awssdk.services.route53.Route53Client;
import software.amazon.awssdk.services.route53.model.RRType;

import java.time.Instant;

public class CreateDistributionTenant {

    public static DistributionTenant
 createDistributionTenantCfHosted(CloudFrontClient cloudFrontClient,
                                                                    Route53Client
 route53Client,
                                                                    String
 distributionId,
                                                                    String domain,
```

```
                                                                    String
hostedZoneId) throws InterruptedException {
       CreateConnectionGroupResponse createConnectionGroupResponse =
cloudFrontClient.createConnectionGroup(builder -> builder
               .ipv6Enabled(true)
               .name("cf-hosted-connection-group")
               .enabled(true));

       route53Client.changeResourceRecordSets(builder -> builder
               .hostedZoneId(hostedZoneId)
               .changeBatch(b1 -> b1
                       .comment("cf-hosted domain validation record")
                       .changes(b2 -> b2
                               .resourceRecordSet(b3 -> b3
                                       .name(domain)
                                       .type(RRType.CNAME)
                                       .ttl(300L)
                                       .resourceRecords(b4 -> b4

.value(createConnectionGroupResponse.connectionGroup().routingEndpoint())))
                               .action("CREATE"))
               ));

       // Give the R53 record time to propagate, if it isn't being returned by
servers yet, the following call will fail
       Thread.sleep(60000);

       CreateDistributionTenantResponse createResponse =
cloudFrontClient.createDistributionTenant(builder -> builder
               .distributionId(distributionId)
               .domains(b1 -> b1
                       .domain(domain))

.connectionGroupId(createConnectionGroupResponse.connectionGroup().id())
               .enabled(false)
               .name("cf-hosted-tenant")
               .parameters(b2 -> b2
                       .name("tenantName")
                       .value("myTenant"))
               .managedCertificateRequest(b3 -> b3
                       .validationTokenHost(ValidationTokenHost.CLOUDFRONT)
               )
       );
```

```
        return createResponse.distributionTenant();
    }

}
```

以下示例演示了如何使用自托管的托管式证书请求来执行此操作。如果您有流量流向您的域，并且无法容忍在迁移期间出现停机，则这是理想之选。在本示例的最后，将在等待域验证和 DNS 设置的状态下创建租户。准备好迁移流量时，请按照 [此处] (https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/managed-cloudfront-certifront-certificates.html#complete-domain-ownership) 完成设置。

```
import software.amazon.awssdk.services.cloudfront.CloudFrontClient;
import
 software.amazon.awssdk.services.cloudfront.model.CreateConnectionGroupResponse;
import
 software.amazon.awssdk.services.cloudfront.model.CreateDistributionTenantResponse;
import software.amazon.awssdk.services.cloudfront.model.DistributionTenant;
import software.amazon.awssdk.services.cloudfront.model.GetConnectionGroupResponse;
import software.amazon.awssdk.services.cloudfront.model.ValidationTokenHost;
import software.amazon.awssdk.services.route53.Route53Client;
import software.amazon.awssdk.services.route53.model.RRType;

import java.time.Instant;

public class CreateDistributionTenant {

    public static DistributionTenant
 createDistributionTenantSelfHosted(CloudFrontClient cloudFrontClient,
                                                                        String
 distributionId,
                                                                        String
 domain) {
        CreateDistributionTenantResponse createResponse =
 cloudFrontClient.createDistributionTenant(builder -> builder
                .distributionId(distributionId)
                .domains(b1 -> b1
                        .domain(domain))
                .parameters(b2 -> b2
                        .name("tenantName")
                        .value("myTenant"))
                .enabled(false)
```

```
                .name("self-hosted-tenant")
                .managedCertificateRequest(b3 -> b3
                        .validationTokenHost(ValidationTokenHost.SELF_HOSTED)
                        .primaryDomainName(domain)
                )
        );

        return createResponse.distributionTenant();
    }

}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - CreateDistribution
  - CreateDistributionTenant

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。

适用于 Java 的 SDK 2.x

> **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。

```
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《AWS SDK for Java 2.x API Reference》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。

适用于 Java 的 SDK 2.x

> **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。

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)
                // .resourceUrlPattern("https://*.example.com/*")  // Optional.
                .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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CloudFrontUtilities中的。

CloudWatch 使用适用于 Java 的 SDK 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

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 基本功能
- 操作
- 场景

开始使用

你好 CloudWatch

以下代码示例展示了如何开始使用 CloudWatch。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListMetrics中的。

基本功能

了解基本功能

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。

适用于 Java 的 SDK 2.x

> **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。

运行演示 CloudWatch 功能的交互式场景。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import
 software.amazon.awssdk.services.cloudwatch.model.DashboardInvalidInputErrorException;
import software.amazon.awssdk.services.cloudwatch.model.DeleteAlarmsResponse;
```

```
import
 software.amazon.awssdk.services.cloudwatch.model.DeleteAnomalyDetectorResponse;
import software.amazon.awssdk.services.cloudwatch.model.DeleteDashboardsResponse;
import software.amazon.awssdk.services.cloudwatch.model.Dimension;
import software.amazon.awssdk.services.cloudwatch.model.GetMetricStatisticsResponse;
import software.amazon.awssdk.services.cloudwatch.model.LimitExceededException;
import software.amazon.awssdk.services.cloudwatch.model.PutDashboardResponse;
import software.amazon.awssdk.services.cloudwatch.model.PutMetricDataResponse;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * To enable billing metrics and statistics for this example, make sure billing
 * alerts are enabled for your account:
 * https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/
monitor_estimated_charges_with_cloudwatch.html#turning_on_billing_metrics
 *
 * This Java code example performs the following tasks:
 *
 * 1. List available namespaces from Amazon CloudWatch.
 * 2. List available metrics within the selected Namespace.
 * 3. Get statistics for the selected metric over the last day.
 * 4. Get CloudWatch estimated billing for the last week.
 * 5. Create a new CloudWatch dashboard with metrics.
 * 6. List dashboards using a paginator.
 * 7. Create a new custom metric by adding data for it.
 * 8. Add the custom metric to the dashboard.
 * 9. Create an alarm for the custom metric.
 * 10. Describe current alarms.
 * 11. Get current data for the new custom metric.
 * 12. Push data into the custom metric to trigger the alarm.
 * 13. Check the alarm state using the action DescribeAlarmsForMetric.
 * 14. Get alarm history for the new alarm.
 * 15. Add an anomaly detector for the custom metric.
 * 16. Describe current anomaly detectors.
```

```
 * 17. Get a metric image for the custom metric.
 * 18. Clean up the Amazon CloudWatch resources.
 */
public class CloudWatchScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    static CloudWatchActions cwActions = new CloudWatchActions();

    private static final Logger logger =
 LoggerFactory.getLogger(CloudWatchScenario.class);
    static Scanner scanner = new Scanner(System.in);
    public static void main(String[] args) throws Throwable {

        final String usage = """

            Usage:
              <myDate> <costDateWeek> <dashboardName> <dashboardJson> <dashboardAdd>
 <settings> <metricImage> \s

            Where:
              myDate - The start date to use to get metric statistics. (For example,
 2023-01-11T18:35:24.00Z.)\s
              costDateWeek - The start date to use to get AWS/Billing statistics.
 (For example, 2023-01-11T18:35:24.00Z.)\s
              dashboardName - The name of the dashboard to create.\s
              dashboardJson - The location of a JSON file to use to create a
 dashboard. (See jsonWidgets.json in javav2/example_code/cloudwatch.)\s
              dashboardAdd - The location of a JSON file to use to update a
 dashboard. (See CloudDashboard.json in javav2/example_code/cloudwatch.)\s
              settings - The location of a JSON file from which various values are
 read. (See settings.json in javav2/example_code/cloudwatch.)\s
              metricImage - The location of a BMP file that is used to create a
 graph.\s
            """;

        if (args.length != 7) {
            logger.info(usage);
            return;
        }
        String myDate = args[0];
        String costDateWeek = args[1];
        String dashboardName = args[2];
        String dashboardJson = args[3];
        String dashboardAdd = args[4];
```

```
        String settings = args[5];
        String metricImage = args[6];

        logger.info(DASHES);
        logger.info("Welcome to the Amazon CloudWatch Basics scenario.");
        logger.info("""
            Amazon CloudWatch is a comprehensive monitoring and observability
service
            provided by Amazon Web Services (AWS). It is designed to help you
monitor your
            AWS resources, applications, and services, as well as on-premises
resources,
            in real-time.

            CloudWatch collects and tracks various types of data, including
metrics,
            logs, and events, from your AWS and on-premises resources. It allows you
to set
            alarms and automatically respond to changes in your environment,
            enabling you to quickly identify and address issues before they impact
your
            applications or services.

            With CloudWatch, you can gain visibility into your entire
infrastructure, from the cloud
            to the edge, and use this information to make informed decisions and
optimize your
            resource utilization.

            This scenario guides you through how to perform Amazon CloudWatch tasks
by using the
            AWS SDK for Java v2. Let's get started...
            """);
        waitForInputToContinue(scanner);

        try {
            runScenario(myDate, costDateWeek, dashboardName, dashboardJson,
dashboardAdd, settings, metricImage);
        } catch (RuntimeException e) {
            e.printStackTrace();
        }
        logger.info(DASHES);
    }
```

```
   private static void runScenario(String myDate, String costDateWeek, String
dashboardName, String dashboardJson, String dashboardAdd, String settings, String
metricImage ) throws Throwable {
       Double dataPoint = Double.parseDouble("10.0");
       logger.info(DASHES);
       logger.info("""
       1. List at least five available unique namespaces from Amazon CloudWatch.
       Select one from the list.
       """);
       String selectedNamespace;
       String selectedMetrics;
       int num;
       try {
           CompletableFuture<ArrayList<String>> future =
cwActions.listNameSpacesAsync();
           ArrayList<String> list = future.join();
           for (int z = 0; z < 5; z++) {
               int index = z + 1;
               logger.info("    " + index + ". {}", list.get(z));
           }

           num = Integer.parseInt(scanner.nextLine());
           if (1 <= num && num <= 5) {
               selectedNamespace = list.get(num - 1);
           } else {
               logger.info("You did not select a valid option.");
               return;
           }
           logger.info("You selected {}", selectedNamespace);

       } catch (RuntimeException rt) {
           Throwable cause = rt.getCause();
           if (cause instanceof CloudWatchException cwEx) {
               logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
           } else {
               logger.info("An unexpected error occurred: " + rt.getMessage());
           }
           throw cause;
       }
       waitForInputToContinue(scanner);
       logger.info(DASHES);

       logger.info(DASHES);
```

```
        logger.info("2. List available metrics within the selected namespace.");
        logger.info("""
            A metric is a measure of the performance or health of your AWS
resources,
            applications, or custom resources. Metrics are the basic building blocks
of CloudWatch
            and provide data points that represent a specific aspect of your system
or application over time.

            Select a metric from the list.
            """);

        Dimension myDimension = null;
        try {
            CompletableFuture<ArrayList<String>> future =
cwActions.listMetsAsync(selectedNamespace);
            ArrayList<String> metList = future.join();
            logger.info("Metrics successfully retrieved. Total metrics: {}",
metList.size());
            for (int z = 0; z < 5; z++) {
                int index = z + 1;
                logger.info("    " + index + ". " + metList.get(z));
            }
            num = Integer.parseInt(scanner.nextLine());
            if (1 <= num && num <= 5) {
                selectedMetrics = metList.get(num - 1);
            } else {
                logger.info("You did not select a valid option.");
                return;
            }
            logger.info("You selected {}", selectedMetrics);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        try {
```

```
            myDimension = cwActions.getSpecificMetAsync(selectedNamespace).join();
            logger.info("Metric statistics successfully retrieved and displayed.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Get statistics for the selected metric over the last day.");
        logger.info("""
            Statistics refer to the various mathematical calculations that can be
performed on the
            collected metrics to derive meaningful insights. Statistics provide a
way to summarize and
            analyze the data collected for a specific metric over a specified time
period.
            """);
        waitForInputToContinue(scanner);
        String metricOption = "";
        ArrayList<String> statTypes = new ArrayList<>();
        statTypes.add("SampleCount");
        statTypes.add("Average");
        statTypes.add("Sum");
        statTypes.add("Minimum");
        statTypes.add("Maximum");

        for (int t = 0; t < 5; t++) {
            logger.info("    " + (t + 1) + ". {}", statTypes.get(t));
        }
        logger.info("Select a metric statistic by entering a number from the
preceding list:");
        num = Integer.parseInt(scanner.nextLine());
        if (1 <= num && num <= 5) {
            metricOption = statTypes.get(num - 1);
        } else {
```

```
            logger.info("You did not select a valid option.");
            return;
        }
        logger.info("You selected " + metricOption);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<GetMetricStatisticsResponse> future =
cwActions.getAndDisplayMetricStatisticsAsync(selectedNamespace, selectedMetrics,
metricOption, myDate, myDimension);
            future.join();
            logger.info("Metric statistics retrieved successfully.");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Get CloudWatch estimated billing for the last week.");
        waitForInputToContinue(scanner);
         try {
            CompletableFuture<GetMetricStatisticsResponse> future =
cwActions.getMetricStatisticsAsync(costDateWeek);
            future.join();

            logger.info("Metric statistics successfully retrieved and displayed.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
             throw cause;
         }
```

```
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Create a new CloudWatch dashboard with metrics.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutDashboardResponse> future =
cwActions.createDashboardWithMetricsAsync(dashboardName, dashboardJson);
            future.join();

        } catch (RuntimeException | IOException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof DashboardInvalidInputErrorException cwEx) {
                logger.info("Invalid CloudWatch data. Error message: {}, Error code
{}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. List dashboards using a paginator.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future = cwActions.listDashboardsAsync();
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);
```

```
        logger.info(DASHES);
        logger.info("7. Create a new custom metric by adding data to it.");
        logger.info("""
            The primary benefit of using a custom metric in Amazon CloudWatch is the
ability to
            monitor and collect data that is specific to your application or
infrastructure.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutMetricDataResponse> future =
cwActions.createNewCustomMetricAsync(dataPoint);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Add an additional metric to the dashboard.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutDashboardResponse> future =
cwActions.addMetricToDashboardAsync(dashboardAdd, dashboardName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof DashboardInvalidInputErrorException cwEx) {
                logger.info("Invalid CloudWatch data. Error message: {}, Error code
{}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
```

```
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("9. Create an alarm for the custom metric.");
        waitForInputToContinue(scanner);
        String alarmName = "" ;
        try {
            CompletableFuture<String> future = cwActions.createAlarmAsync(settings);
            alarmName = future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof LimitExceededException cwEx) {
                logger.info("The quota for alarms has been reached: Error message:
{}, Error code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("10. Describe ten current alarms.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future = cwActions.describeAlarmsAsync();
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);
```

```
        logger.info(DASHES);
        logger.info("11. Get current data for new custom metric.");
        try {
            CompletableFuture<Void> future =
cwActions.getCustomMetricDataAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("12. Push data into the custom metric to trigger the alarm.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutMetricDataResponse> future =
cwActions.addMetricDataForAlarmAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("13. Check the alarm state using the action
DescribeAlarmsForMetric.");
```

```
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
cwActions.checkForMetricAlarmAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("14. Get alarm history for the new alarm.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
cwActions.getAlarmHistoryAsync(settings, myDate);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("15. Add an anomaly detector for the custom metric.");
        logger.info("""
            An anomaly detector is a feature that automatically detects unusual
patterns or deviations in your
```

```
            monitored metrics. It uses machine learning algorithms to analyze the
historical behavior
            of your metrics and establish a baseline.

            The anomaly detector then compares the current metric values against
this baseline and
            identifies any anomalies or outliers that may indicate potential issues
or unexpected changes
            in your system's performance or behavior.

            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
cwActions.addAnomalyDetectorAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("16. Describe current anomaly detectors.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
cwActions.describeAnomalyDetectorsAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
```

```
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("17. Get a metric image for the custom metric.");
        try {
            CompletableFuture<Void> future =
cwActions.downloadAndSaveMetricImageAsync(metricImage);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("18. Clean up the Amazon CloudWatch resources.");

        try {
            logger.info(". Delete the Dashboard.");
            waitForInputToContinue(scanner);
            CompletableFuture<DeleteDashboardsResponse> future =
cwActions.deleteDashboardAsync(dashboardName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
```

```
            throw cause;
        }

        try {
            logger.info("Delete the alarm.");
            waitForInputToContinue(scanner);
            CompletableFuture<DeleteAlarmsResponse> future =
cwActions.deleteCWAlarmAsync(alarmName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        try {
            logger.info("Delete the anomaly detector.");
            waitForInputToContinue(scanner);
            CompletableFuture<DeleteAnomalyDetectorResponse> future =
cwActions.deleteAnomalyDetectorAsync(settings);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof CloudWatchException cwEx) {
                logger.info("CloudWatch error occurred: Error message: {}, Error
code {}", cwEx.getMessage(), cwEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("The Amazon CloudWatch example scenario is complete.");
        logger.info(DASHES);
```

```
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();
            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

CloudWatch SDK 方法的包装器类。

```
public class CloudWatchActions {

    private static CloudWatchAsyncClient cloudWatchAsyncClient;

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

    /**
     * Retrieves an asynchronous CloudWatch client instance.
     *
     * <p>
     * This method ensures that the CloudWatch client is initialized with the
 following configurations:
     * <ul>
     *     <li>Maximum concurrency: 100</li>
     *     <li>Connection timeout: 60 seconds</li>
     *     <li>Read timeout: 60 seconds</li>
     *     <li>Write timeout: 60 seconds</li>
     *     <li>API call timeout: 2 minutes</li>
     *     <li>API call attempt timeout: 90 seconds</li>
     *     <li>Retry strategy: STANDARD</li>
```

```
     * </ul>
     * </p>
     *
     * @return the asynchronous CloudWatch client instance
     */
    private static CloudWatchAsyncClient getAsyncClient() {
        if (cloudWatchAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            cloudWatchAsyncClient = CloudWatchAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return cloudWatchAsyncClient;
    }

    /**
     * Deletes an Anomaly Detector.
     *
     * @param fileName the name of the file containing the Anomaly Detector
configuration
     * @return a CompletableFuture that represents the asynchronous deletion of the
Anomaly Detector
     */
    public CompletableFuture<DeleteAnomalyDetectorResponse>
deleteAnomalyDetectorAsync(String fileName) {
        CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
```

```
                return new ObjectMapper().readTree(parser); // Return the root node
            } catch (IOException e) {
                throw new RuntimeException("Failed to read or parse the file", e);
            }
        });

        return readFileFuture.thenCompose(rootNode -> {
            String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
            String customMetricName =
rootNode.findValue("customMetricName").asText();

            SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                .metricName(customMetricName)
                .namespace(customMetricNamespace)
                .stat("Maximum")
                .build();

            DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
                .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                .build();

            return getAsyncClient().deleteAnomalyDetector(request);
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Failed to delete the Anomaly Detector",
exception);
            } else {
                logger.info("Successfully deleted the Anomaly Detector.");
            }
        });
    }

    /**
     * Deletes a CloudWatch alarm.
     *
     * @param alarmName the name of the alarm to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous operation
to delete the alarm
     * the {@link DeleteAlarmsResponse} is returned when the operation completes
successfully,
     * or a {@link RuntimeException} is thrown if the operation fails
```

```
     */
    public CompletableFuture<DeleteAlarmsResponse> deleteCWAlarmAsync(String
alarmName) {
        DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
            .alarmNames(alarmName)
            .build();

        return getAsyncClient().deleteAlarms(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to delete the alarm:{} " +
alarmName, exception);
                } else {
                    logger.info("Successfully deleted alarm {} ", alarmName);
                }
            });
    }

    /**
     * Deletes the specified dashboard.
     *
     * @param dashboardName the name of the dashboard to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous operation
of deleting the dashboard
     * @throws RuntimeException if the dashboard deletion fails
     */
    public CompletableFuture<DeleteDashboardsResponse> deleteDashboardAsync(String
dashboardName) {
        DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
            .dashboardNames(dashboardName)
            .build();

        return getAsyncClient().deleteDashboards(dashboardsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to delete the dashboard: " +
dashboardName, exception);
                } else {
                    logger.info("{} was successfully deleted.", dashboardName);
                }
            });
    }
```

```
   /**
    * Retrieves and saves a custom metric image to a file.
    *
    * @param fileName the name of the file to save the metric image to
    * @return a {@link CompletableFuture} that completes when the image has been
saved to the file
    */
   public CompletableFuture<Void> downloadAndSaveMetricImageAsync(String fileName)
{
       logger.info("Getting Image data for custom metric.");
       String myJSON = """
             {
                 "title": "Example Metric Graph",
                 "view": "timeSeries",
                 "stacked ": false,
                 "period": 10,
                 "width": 1400,
                 "height": 600,
                 "metrics": [
                     [
                     "AWS/Billing",
                     "EstimatedCharges",
                     "Currency",
                     "USD"
                    ]
                 ]
             }
           """;

       GetMetricWidgetImageRequest imageRequest =
GetMetricWidgetImageRequest.builder()
           .metricWidget(myJSON)
           .build();

       return getAsyncClient().getMetricWidgetImage(imageRequest)
           .thenCompose(response -> {
               SdkBytes sdkBytes = response.metricWidgetImage();
               byte[] bytes = sdkBytes.asByteArray();
               return CompletableFuture.runAsync(() -> {
                   try {
                       File outputFile = new File(fileName);
                       try (FileOutputStream outputStream = new
FileOutputStream(outputFile)) {
```

```
                            outputStream.write(bytes);
                        }
                    } catch (IOException e) {
                        throw new RuntimeException("Failed to write image to file",
e);
                    }
                });
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Error getting and saving metric
image", exception);
                } else {
                    logger.info("Image data saved successfully to {}", fileName);
                }
            });
    }


    /**
     * Describes the anomaly detectors based on the specified JSON file.
     *
     * @param fileName the name of the JSON file containing the custom metric
namespace and name
     * @return a {@link CompletableFuture} that completes when the anomaly detectors
have been described
     * @throws RuntimeException if there is a failure during the operation, such as
when reading or parsing the JSON file,
     *                          or when describing the anomaly detectors
     */
    public CompletableFuture<Void> describeAnomalyDetectorsAsync(String fileName) {
        CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                return new ObjectMapper().readTree(parser);
            } catch (IOException e) {
                throw new RuntimeException("Failed to read or parse the file", e);
            }
        });

        return readFileFuture.thenCompose(rootNode -> {
            try {
```

```
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();

                DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
                    .maxResults(10)
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

                return
getAsyncClient().describeAnomalyDetectors(detectorsRequest).thenAccept(response ->
{
                    List<AnomalyDetector> anomalyDetectorList =
response.anomalyDetectors();
                    for (AnomalyDetector detector : anomalyDetectorList) {
                        logger.info("Metric name: {} ",
detector.singleMetricAnomalyDetector().metricName());
                        logger.info("State: {} ", detector.stateValue());
                    }
                });
            } catch (RuntimeException e) {
                throw new RuntimeException("Failed to describe anomaly detectors",
e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error describing anomaly detectors",
exception);
            }
        });
    }


    /**
     * Adds an anomaly detector for the given file.
     *
     * @param fileName the name of the file containing the anomaly detector
configuration
     * @return a {@link CompletableFuture} that completes when the anomaly detector
has been added
     */
```

```
   public CompletableFuture<Void> addAnomalyDetectorAsync(String fileName) {
       CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               return new ObjectMapper().readTree(parser); // Return the root node
           } catch (IOException e) {
               throw new RuntimeException("Failed to read or parse the file", e);
           }
       });

       return readFileFuture.thenCompose(rootNode -> {
           try {
               String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
               String customMetricName =
rootNode.findValue("customMetricName").asText();

               SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                   .metricName(customMetricName)
                   .namespace(customMetricNamespace)
                   .stat("Maximum")
                   .build();

               PutAnomalyDetectorRequest anomalyDetectorRequest =
PutAnomalyDetectorRequest.builder()
                   .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                   .build();

               return
getAsyncClient().putAnomalyDetector(anomalyDetectorRequest).thenAccept(response ->
{
                   logger.info("Added anomaly detector for metric {}",
customMetricName);
               });
           } catch (Exception e) {
               throw new RuntimeException("Failed to create anomaly detector", e);
           }
       }).whenComplete((result, exception) -> {
           if (exception != null) {
               throw new RuntimeException("Error adding anomaly detector",
exception);
```

```
            }
        });
    }


    /**
     * Retrieves the alarm history for a given alarm name and date range.
     *
     * @param fileName the path to the JSON file containing the alarm name
     * @param date     the date to start the alarm history search (in the format
"yyyy-MM-dd'T'HH:mm:ss'Z'")
     * @return a {@code CompletableFuture<Void>} that completes when the alarm
history has been retrieved and processed
     */
    public CompletableFuture<Void> getAlarmHistoryAsync(String fileName, String
date) {
        CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
                return rootNode.findValue("exampleAlarmName").asText(); // Return
alarmName from the JSON file
            } catch (IOException e) {
                throw new RuntimeException("Failed to read or parse the file", e);
            }
        });

        // Use the alarm name to describe alarm history with a paginator.
        return readFileFuture.thenCompose(alarmName -> {
            try {
                Instant start = Instant.parse(date);
                Instant endDate = Instant.now();
                DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
                    .startDate(start)
                    .endDate(endDate)
                    .alarmName(alarmName)
                    .historyItemType(HistoryItemType.ACTION)
                    .build();

                // Use the paginator to paginate through alarm history pages.
```

```
                DescribeAlarmHistoryPublisher historyPublisher =
getAsyncClient().describeAlarmHistoryPaginator(historyRequest);
                CompletableFuture<Void> future = historyPublisher
                    .subscribe(response -> response.alarmHistoryItems().forEach(item
-> {
                        logger.info("History summary: {}", item.historySummary());
                        logger.info("Timestamp: {}", item.timestamp());
                    }))
                    .whenComplete((result, exception) -> {
                        if (exception != null) {
                            logger.error("Error occurred while getting alarm
history: " + exception.getMessage(), exception);
                        } else {
                            logger.info("Successfully retrieved all alarm
history.");
                        }
                    });

                // Return the future to the calling code for further handling
                return future;
            } catch (Exception e) {
                throw new RuntimeException("Failed to process alarm history", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error completing alarm history
processing", exception);
            }
        });
    }



    /**
     * Checks for a metric alarm in AWS CloudWatch.
     *
     * @param fileName the name of the file containing the JSON configuration for
the custom metric
     * @return a {@link CompletableFuture} that completes when the check for the
metric alarm is complete
     */
    public CompletableFuture<Void> checkForMetricAlarmAsync(String fileName) {
        CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
```

```
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
                return rootNode.toString(); // Return JSON as a string for further
processing
            } catch (IOException e) {
                throw new RuntimeException("Failed to read file", e);
            }
        });

        return readFileFuture.thenCompose(jsonContent -> {
            try {
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();

                DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

                return checkForAlarmAsync(metricRequest, customMetricName, 10);

            } catch (IOException e) {
                throw new RuntimeException("Failed to parse JSON content", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error checking metric alarm",
exception);
            }
        });
    }

    // Recursive method to check for the alarm.

    /**
     * Checks for the existence of an alarm asynchronously for the specified metric.
```

```
    *
    * @param metricRequest    the request to describe the alarms for the specified
metric
    * @param customMetricName the name of the custom metric to check for an alarm
    * @param retries          the number of retries to perform if no alarm is found
    * @return a {@link CompletableFuture} that completes when an alarm is found or
the maximum number of retries has been reached
    */
   private static CompletableFuture<Void>
checkForAlarmAsync(DescribeAlarmsForMetricRequest metricRequest, String
customMetricName, int retries) {
       if (retries == 0) {
           return CompletableFuture.completedFuture(null).thenRun(() ->
               logger.info("No Alarm state found for {} after 10 retries.",
customMetricName)
           );
       }

       return
(getAsyncClient().describeAlarmsForMetric(metricRequest).thenCompose(response -> {
           if (response.hasMetricAlarms()) {
               logger.info("Alarm state found for {}", customMetricName);
               return CompletableFuture.completedFuture(null); // Alarm found,
complete the future
           } else {
               return CompletableFuture.runAsync(() -> {
                   try {
                       Thread.sleep(20000);
                       logger.info(".");
                   } catch (InterruptedException e) {
                       throw new RuntimeException("Interrupted while waiting to
retry", e);
                   }
               }).thenCompose(v -> checkForAlarmAsync(metricRequest,
customMetricName, retries - 1)); // Recursive call
           }
       }));
   }


   /**
    * Adds metric data for an alarm asynchronously.
    *
    * @param fileName the name of the JSON file containing the metric data
```

```
    * @return a CompletableFuture that asynchronously returns the
PutMetricDataResponse
    */
   public CompletableFuture<PutMetricDataResponse>
addMetricDataForAlarmAsync(String fileName) {
       CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
               return rootNode.toString(); // Return JSON as a string for further
processing
           } catch (IOException e) {
               throw new RuntimeException("Failed to read file", e);
           }
       });

       return readFileFuture.thenCompose(jsonContent -> {
           try {
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
               String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
               String customMetricName =
rootNode.findValue("customMetricName").asText();
               Instant instant = Instant.now();

               // Create MetricDatum objects.
               MetricDatum datum1 = MetricDatum.builder()
                   .metricName(customMetricName)
                   .unit(StandardUnit.NONE)
                   .value(1001.00)
                   .timestamp(instant)
                   .build();

               MetricDatum datum2 = MetricDatum.builder()
                   .metricName(customMetricName)
                   .unit(StandardUnit.NONE)
                   .value(1002.00)
                   .timestamp(instant)
                   .build();
```

```
                List<MetricDatum> metricDataList = new ArrayList<>();
                metricDataList.add(datum1);
                metricDataList.add(datum2);

                // Build the PutMetricData request.
                PutMetricDataRequest request = PutMetricDataRequest.builder()
                    .namespace(customMetricNamespace)
                    .metricData(metricDataList)
                    .build();

                // Send the request asynchronously.
                return getAsyncClient().putMetricData(request);

            } catch (IOException e) {
                CompletableFuture<PutMetricDataResponse> failedFuture = new
CompletableFuture<>();
                failedFuture.completeExceptionally(new RuntimeException("Failed to
parse JSON content", e));
                return failedFuture;
            }
        }).whenComplete((response, exception) -> {
            if (exception != null) {
                logger.error("Failed to put metric data: " + exception.getMessage(),
exception);
            } else {
                logger.info("Added metric values for metric.");
            }
        });
    }


    /**
     * Retrieves custom metric data from the AWS CloudWatch service.
     *
     * @param fileName the name of the file containing the custom metric information
     * @return a {@link CompletableFuture} that completes when the metric data has
been retrieved
     */
    public CompletableFuture<Void> getCustomMetricDataAsync(String fileName) {
        CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
            try {
                // Read values from the JSON file.
```

```
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
                return rootNode.toString(); // Return JSON as a string for further
processing
            } catch (IOException e) {
                throw new RuntimeException("Failed to read file", e);
            }
        });

        return readFileFuture.thenCompose(jsonContent -> {
            try {
                // Parse the JSON string to extract relevant values.
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();

                // Set the current time and date range for metric query.
                Instant nowDate = Instant.now();
                long hours = 1;
                long minutes = 30;
                Instant endTime = nowDate.plus(hours,
ChronoUnit.HOURS).plus(minutes, ChronoUnit.MINUTES);

                Metric met = Metric.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

                MetricStat metStat = MetricStat.builder()
                    .stat("Maximum")
                    .period(60)  // Assuming period in seconds
                    .metric(met)
                    .build();

                MetricDataQuery dataQuery = MetricDataQuery.builder()
                    .metricStat(metStat)
                    .id("foo2")
                    .returnData(true)
                    .build();
```

```
                List<MetricDataQuery> dq = new ArrayList<>();
                dq.add(dataQuery);

                GetMetricDataRequest getMetricDataRequest =
GetMetricDataRequest.builder()
                    .maxDatapoints(10)
                    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
                    .startTime(nowDate)
                    .endTime(endTime)
                    .metricDataQueries(dq)
                    .build();

                // Call the async method for CloudWatch data retrieval.
                return getAsyncClient().getMetricData(getMetricDataRequest);

            } catch (IOException e) {
                throw new RuntimeException("Failed to parse JSON content", e);
            }
        }).thenAccept(response -> {
            List<MetricDataResult> data = response.metricDataResults();
            for (MetricDataResult item : data) {
                logger.info("The label is: {}", item.label());
                logger.info("The status code is: {}", item.statusCode().toString());
            }
        }).exceptionally(exception -> {
            throw new RuntimeException("Failed to get metric data", exception);
        });
    }


    /**
     * Describes the CloudWatch alarms of the 'METRIC_ALARM' type.
     *
     * @return a {@link CompletableFuture} that represents the asynchronous
operation
     * of describing the CloudWatch alarms. The future completes when the
     * operation is finished, either successfully or with an error.
     */
    public CompletableFuture<Void> describeAlarmsAsync() {
        List<AlarmType> typeList = new ArrayList<>();
        typeList.add(AlarmType.METRIC_ALARM);
        DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
            .alarmTypes(typeList)
```

```
            .maxRecords(10)
            .build();

        return getAsyncClient().describeAlarms(alarmsRequest)
            .thenAccept(response -> {
                List<MetricAlarm> alarmList = response.metricAlarms();
                for (MetricAlarm alarm : alarmList) {
                    logger.info("Alarm name: {}", alarm.alarmName());
                    logger.info("Alarm description: {} ", alarm.alarmDescription());
                }
            })
            .whenComplete((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to describe alarms: {}", ex.getMessage());
                } else {
                    logger.info("Successfully described alarms.");
                }
            });
    }

    /**
     * Creates an alarm based on the configuration provided in a JSON file.
     *
     * @param fileName the name of the JSON file containing the alarm configuration
     * @return a CompletableFuture that represents the asynchronous operation of
creating the alarm
     * @throws RuntimeException if an exception occurs while reading the JSON file
or creating the alarm
     */
    public CompletableFuture<String> createAlarmAsync(String fileName) {
        com.fasterxml.jackson.databind.JsonNode rootNode;
        try {
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            rootNode = new ObjectMapper().readTree(parser);
        } catch (IOException e) {
            throw new RuntimeException("Failed to read the alarm configuration
file", e);
        }

        // Extract values from the JSON node.
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName = rootNode.findValue("customMetricName").asText();
        String alarmName = rootNode.findValue("exampleAlarmName").asText();
```

```
        String emailTopic = rootNode.findValue("emailTopic").asText();
        String accountId = rootNode.findValue("accountId").asText();
        String region = rootNode.findValue("region").asText();

        // Create a List for alarm actions.
        List<String> alarmActions = new ArrayList<>();
        alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);

        PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
            .alarmActions(alarmActions)
            .alarmDescription("Example metric alarm")
            .alarmName(alarmName)

.comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
            .threshold(100.00)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .evaluationPeriods(1)
            .period(10)
            .statistic("Maximum")
            .datapointsToAlarm(1)
            .treatMissingData("ignore")
            .build();

        // Call the putMetricAlarm asynchronously and handle the result.
        return getAsyncClient().putMetricAlarm(alarmRequest)
            .handle((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to create alarm: {}", ex.getMessage());
                    throw new RuntimeException("Failed to create alarm", ex);
                } else {
                    logger.info("{} was successfully created!", alarmName);
                    return alarmName;
                }
            });
    }

    /**
     * Adds a metric to a dashboard asynchronously.
     *
     * @param fileName      the name of the file containing the dashboard content
     * @param dashboardName the name of the dashboard to be updated
```

```
    * @return a {@link CompletableFuture} representing the asynchronous operation,
which will complete with a
    * {@link PutDashboardResponse} when the dashboard is successfully updated
    */
   public CompletableFuture<PutDashboardResponse> addMetricToDashboardAsync(String
fileName, String dashboardName) {
       String dashboardBody;
       try {
           dashboardBody = readFileAsString(fileName);
       } catch (IOException e) {
           throw new RuntimeException("Failed to read the dashboard file", e);
       }

       PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
           .dashboardName(dashboardName)
           .dashboardBody(dashboardBody)
           .build();

       return getAsyncClient().putDashboard(dashboardRequest)
           .handle((response, ex) -> {
               if (ex != null) {
                   logger.info("Failed to update dashboard: {}", ex.getMessage());
                   throw new RuntimeException("Error updating dashboard", ex);
               } else {
                   logger.info("{} was successfully updated.", dashboardName);
                   return response;
               }
           });
   }

   /**
    * Creates a new custom metric.
    *
    * @param dataPoint the data point to be added to the custom metric
    * @return a {@link CompletableFuture} representing the asynchronous operation
of adding the custom metric
    */
   public CompletableFuture<PutMetricDataResponse>
createNewCustomMetricAsync(Double dataPoint) {
       Dimension dimension = Dimension.builder()
           .name("UNIQUE_PAGES")
           .value("URLS")
           .build();
```

```
        // Set an Instant object for the current time in UTC.
        String time =
ZonedDateTime.now(ZoneOffset.UTC).format(DateTimeFormatter.ISO_INSTANT);
        Instant instant = Instant.parse(time);

        // Create the MetricDatum.
        MetricDatum datum = MetricDatum.builder()
            .metricName("PAGES_VISITED")
            .unit(StandardUnit.NONE)
            .value(dataPoint)
            .timestamp(instant)
            .dimensions(dimension)
            .build();

        PutMetricDataRequest request = PutMetricDataRequest.builder()
            .namespace("SITE/TRAFFIC")
            .metricData(datum)
            .build();

        return getAsyncClient().putMetricData(request)
            .whenComplete((response, ex) -> {
                if (ex != null) {
                    throw new RuntimeException("Error adding custom metric", ex);
                } else {
                    logger.info("Successfully added metric values for
PAGES_VISITED.");
                }
            });
    }

    /**
     * Lists the available dashboards.
     *
     * @return a {@link CompletableFuture} that completes when the operation is
finished.
     * The future will complete exceptionally if an error occurs while listing the
dashboards.
     */
    public CompletableFuture<Void> listDashboardsAsync() {
        ListDashboardsRequest listDashboardsRequest =
ListDashboardsRequest.builder().build();
        ListDashboardsPublisher paginator =
getAsyncClient().listDashboardsPaginator(listDashboardsRequest);
        return paginator.subscribe(response -> {
```

```
            response.dashboardEntries().forEach(entry -> {
                logger.info("Dashboard name is: {} ", entry.dashboardName());
                logger.info("Dashboard ARN is: {} ", entry.dashboardArn());
            });
        }).exceptionally(ex -> {
            logger.info("Failed to list dashboards: {} ", ex.getMessage());
            throw new RuntimeException("Error occurred while listing dashboards",
ex);
        });
    }


    /**
     * Creates a new dashboard with the specified name and metrics from the given
file.
     *
     * @param dashboardName the name of the dashboard to be created
     * @param fileName      the name of the file containing the dashboard body
     * @return a {@link CompletableFuture} representing the asynchronous operation
of creating the dashboard
     * @throws IOException if there is an error reading the dashboard body from the
file
     */
    public CompletableFuture<PutDashboardResponse>
createDashboardWithMetricsAsync(String dashboardName, String fileName) throws
IOException {
        String dashboardBody = readFileAsString(fileName);
        PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
            .dashboardName(dashboardName)
            .dashboardBody(dashboardBody)
            .build();

        return getAsyncClient().putDashboard(dashboardRequest)
            .handle((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to create dashboard: {}", ex.getMessage());
                    throw new RuntimeException("Dashboard creation failed", ex);
                } else {
                    // Handle the normal response case
                    logger.info("{} was successfully created.", dashboardName);
                    List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
                    if (messages.isEmpty()) {
                        logger.info("There are no messages in the new Dashboard.");
```

```
                    } else {
                        for (DashboardValidationMessage message : messages) {
                            logger.info("Message: {}", message.message());
                        }
                    }
                    return response; // Return the response for further use
                }
            });
    }


    /**
     * Retrieves the metric statistics for the "EstimatedCharges" metric in the
"AWS/Billing" namespace.
     *
     * @param costDateWeek the start date for the metric statistics, in the format
of an ISO-8601 date string (e.g., "2023-04-05")
     * @return a {@link CompletableFuture} that, when completed, contains the {@link
GetMetricStatisticsResponse} with the retrieved metric statistics
     * @throws RuntimeException if the metric statistics cannot be retrieved
successfully
     */
    public CompletableFuture<GetMetricStatisticsResponse>
getMetricStatisticsAsync(String costDateWeek) {
        Instant start = Instant.parse(costDateWeek);
        Instant endDate = Instant.now();

        // Define dimension
        Dimension dimension = Dimension.builder()
            .name("Currency")
            .value("USD")
            .build();

        List<Dimension> dimensionList = new ArrayList<>();
        dimensionList.add(dimension);

        GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
            .metricName("EstimatedCharges")
            .namespace("AWS/Billing")
            .dimensions(dimensionList)
            .statistics(Statistic.MAXIMUM)
            .startTime(start)
            .endTime(endDate)
```

```
            .period(86400) // One day period
            .build();

        return getAsyncClient().getMetricStatistics(statisticsRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    List<Datapoint> data = response.datapoints();
                    if (!data.isEmpty()) {
                        for (Datapoint datapoint : data) {
                            logger.info("Timestamp: {} Maximum value: {})",
datapoint.timestamp(), datapoint.maximum());
                        }
                    } else {
                        logger.info("The returned data list is empty");
                    }
                } else {
                    throw new RuntimeException("Failed to get metric statistics: " +
exception.getMessage(), exception);
                }
            });
    }


    /**
     * Retrieves and displays metric statistics for the specified parameters.
     *
     * @param nameSpace    the namespace for the metric
     * @param metVal       the name of the metric
     * @param metricOption the statistic to retrieve for the metric (e.g.,
"Maximum", "Average")
     * @param date         the date for which to retrieve the metric statistics, in
the format "yyyy-MM-dd'T'HH:mm:ss'Z'"
     * @param myDimension  the dimension(s) to filter the metric statistics by
     * @return a {@link CompletableFuture} that completes when the metric statistics
have been retrieved and displayed
     */
    public CompletableFuture<GetMetricStatisticsResponse>
getAndDisplayMetricStatisticsAsync(String nameSpace, String metVal,

         String metricOption, String date, Dimension myDimension) {

        Instant start = Instant.parse(date);
        Instant endDate = Instant.now();
```

```
        // Building the request for metric statistics.
        GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
            .endTime(endDate)
            .startTime(start)
            .dimensions(myDimension)
            .metricName(metVal)
            .namespace(nameSpace)
            .period(86400) // 1 day period
            .statistics(Statistic.fromValue(metricOption))
            .build();

        return getAsyncClient().getMetricStatistics(statisticsRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    List<Datapoint> data = response.datapoints();
                    if (!data.isEmpty()) {
                        for (Datapoint datapoint : data) {
                            logger.info("Timestamp: {} Maximum value: {}",
datapoint.timestamp(), datapoint.maximum());
                        }
                    } else {
                        logger.info("The returned data list is empty");
                    }
                } else {
                    logger.info("Failed to get metric statistics: {} ",
exception.getMessage());
                }
            })
            .exceptionally(exception -> {
                throw new RuntimeException("Error while getting metric statistics: "
+ exception.getMessage(), exception);
            });
    }


    /**
     * Retrieves a list of metric names for the specified namespace.
     *
     * @param namespace the namespace for which to retrieve the metric names
     * @return a {@link CompletableFuture} that, when completed, contains an {@link
ArrayList} of
     * the metric names in the specified namespace
     * @throws RuntimeException if an error occurs while listing the metrics
```

```
     */
    public CompletableFuture<ArrayList<String>> listMetsAsync(String namespace) {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        ListMetricsPublisher metricsPaginator =
getAsyncClient().listMetricsPaginator(request);
        Set<String> metSet = new HashSet<>();
        CompletableFuture<Void> future = metricsPaginator.subscribe(response -> {
            response.metrics().forEach(metric -> {
                String metricName = metric.metricName();
                metSet.add(metricName);
            });
        });

        return future
            .thenApply(ignored -> new ArrayList<>(metSet))
            .exceptionally(exception -> {
                throw new RuntimeException("Failed to list metrics: " +
exception.getMessage(), exception);
            });
    }

    /**
     * Lists the available namespaces for the current AWS account.
     *
     * @return a {@link CompletableFuture} that, when completed, contains an {@link
ArrayList} of the available namespace names.
     * @throws RuntimeException if an error occurs while listing the namespaces.
     */
    public CompletableFuture<ArrayList<String>> listNameSpacesAsync() {
        ArrayList<String> nameSpaceList = new ArrayList<>();
        ListMetricsRequest request = ListMetricsRequest.builder().build();

        ListMetricsPublisher metricsPaginator =
getAsyncClient().listMetricsPaginator(request);
        CompletableFuture<Void> future = metricsPaginator.subscribe(response -> {
            response.metrics().forEach(metric -> {
                String namespace = metric.namespace();
                if (!nameSpaceList.contains(namespace)) {
                    nameSpaceList.add(namespace);
                }
            });
```

```
        });

        return future
            .thenApply(ignored -> nameSpaceList)
            .exceptionally(exception -> {
                throw new RuntimeException("Failed to list namespaces: " +
exception.getMessage(), exception);
            });
    }
    /**
     * Retrieves the specific metric asynchronously.
     *
     * @param namespace the namespace of the metric to retrieve
     * @return a CompletableFuture that completes with the first dimension of the
first metric found in the specified namespace,
     * or throws a RuntimeException if an error occurs or no metrics or dimensions
are found
     */
    public CompletableFuture<Dimension> getSpecificMetAsync(String namespace) {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        return getAsyncClient().listMetrics(request).handle((response, exception) ->
{
            if (exception != null) {
                logger.info("Error occurred while listing metrics: {} ",
exception.getMessage());
                throw new RuntimeException("Failed to retrieve specific metric
dimension", exception);
            } else {
                List<Metric> myList = response.metrics();
                if (!myList.isEmpty()) {
                    Metric metric = myList.get(0);
                    if (!metric.dimensions().isEmpty()) {
                        return metric.dimensions().get(0); // Return the first
dimension
                    }
                }
                throw new RuntimeException("No metrics or dimensions found");
            }
        });
    }
```

```
    public static String readFileAsString(String file) throws IOException {
        return new String(Files.readAllBytes(Paths.get(file)));
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - DeleteAlarms
  - DeleteAnomalyDetector
  - DeleteDashboards
  - DescribeAlarmHistory
  - DescribeAlarms
  - DescribeAlarmsForMetric
  - DescribeAnomalyDetectors
  - GetMetricData
  - GetMetricStatistics
  - GetMetricWidgetImage
  - ListMetrics
  - PutAnomalyDetector
  - PutDashboard
  - PutMetricAlarm
  - PutMetricData

操作

`DeleteAlarms`

以下代码示例演示了如何使用 `DeleteAlarms`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Deletes a CloudWatch alarm.
     *
     * @param alarmName the name of the alarm to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous operation
to delete the alarm
     * the {@link DeleteAlarmsResponse} is returned when the operation completes
successfully,
     * or a {@link RuntimeException} is thrown if the operation fails
     */
    public CompletableFuture<DeleteAlarmsResponse> deleteCWAlarmAsync(String
alarmName) {
        DeleteAlarmsRequest request = DeleteAlarmsRequest.builder()
            .alarmNames(alarmName)
            .build();

        return getAsyncClient().deleteAlarms(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to delete the alarm:{} " +
alarmName, exception);
                } else {
                    logger.info("Successfully deleted alarm {} ", alarmName);
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteAlarms中的。

`DeleteAnomalyDetector`

以下代码示例演示了如何使用 `DeleteAnomalyDetector`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Deletes an Anomaly Detector.
    *
    * @param fileName the name of the file containing the Anomaly Detector
configuration
    * @return a CompletableFuture that represents the asynchronous deletion of the
Anomaly Detector
    */
   public CompletableFuture<DeleteAnomalyDetectorResponse>
deleteAnomalyDetectorAsync(String fileName) {
       CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               return new ObjectMapper().readTree(parser); // Return the root node
           } catch (IOException e) {
               throw new RuntimeException("Failed to read or parse the file", e);
           }
       });

       return readFileFuture.thenCompose(rootNode -> {
           String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
           String customMetricName =
rootNode.findValue("customMetricName").asText();

           SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
               .metricName(customMetricName)
               .namespace(customMetricNamespace)
               .stat("Maximum")
               .build();

           DeleteAnomalyDetectorRequest request =
DeleteAnomalyDetectorRequest.builder()
               .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
               .build();

           return getAsyncClient().deleteAnomalyDetector(request);
       }).whenComplete((result, exception) -> {
           if (exception != null) {
```

```
                throw new RuntimeException("Failed to delete the Anomaly Detector",
exception);
            } else {
                logger.info("Successfully deleted the Anomaly Detector.");
            }
        });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteAnomalyDetector中的。

`DeleteDashboards`

以下代码示例演示了如何使用 `DeleteDashboards`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Deletes the specified dashboard.
     *
     * @param dashboardName the name of the dashboard to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous operation
of deleting the dashboard
     * @throws RuntimeException if the dashboard deletion fails
     */
    public CompletableFuture<DeleteDashboardsResponse> deleteDashboardAsync(String
dashboardName) {
        DeleteDashboardsRequest dashboardsRequest =
DeleteDashboardsRequest.builder()
            .dashboardNames(dashboardName)
            .build();

        return getAsyncClient().deleteDashboards(dashboardsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
```

```
                    throw new RuntimeException("Failed to delete the dashboard: " +
dashboardName, exception);
                } else {
                    logger.info("{} was successfully deleted.", dashboardName);
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteDashboards中的。

`DescribeAlarmHistory`

以下代码示例演示了如何使用 `DescribeAlarmHistory`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Retrieves the alarm history for a given alarm name and date range.
     *
     * @param fileName the path to the JSON file containing the alarm name
     * @param date     the date to start the alarm history search (in the format
"yyyy-MM-dd'T'HH:mm:ss'Z'")
     * @return a {@code CompletableFuture<Void>} that completes when the alarm
history has been retrieved and processed
     */
    public CompletableFuture<Void> getAlarmHistoryAsync(String fileName, String
date) {
        CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
```

```
                return rootNode.findValue("exampleAlarmName").asText(); // Return
alarmName from the JSON file
            } catch (IOException e) {
                throw new RuntimeException("Failed to read or parse the file", e);
            }
        });

        // Use the alarm name to describe alarm history with a paginator.
        return readFileFuture.thenCompose(alarmName -> {
            try {
                Instant start = Instant.parse(date);
                Instant endDate = Instant.now();
                DescribeAlarmHistoryRequest historyRequest =
DescribeAlarmHistoryRequest.builder()
                    .startDate(start)
                    .endDate(endDate)
                    .alarmName(alarmName)
                    .historyItemType(HistoryItemType.ACTION)
                    .build();

                // Use the paginator to paginate through alarm history pages.
                DescribeAlarmHistoryPublisher historyPublisher =
getAsyncClient().describeAlarmHistoryPaginator(historyRequest);
                CompletableFuture<Void> future = historyPublisher
                    .subscribe(response -> response.alarmHistoryItems().forEach(item
-> {
                        logger.info("History summary: {}", item.historySummary());
                        logger.info("Timestamp: {}", item.timestamp());
                    }))
                    .whenComplete((result, exception) -> {
                        if (exception != null) {
                            logger.error("Error occurred while getting alarm
history: " + exception.getMessage(), exception);
                        } else {
                            logger.info("Successfully retrieved all alarm
history.");
                        }
                    });

                // Return the future to the calling code for further handling
                return future;
            } catch (Exception e) {
                throw new RuntimeException("Failed to process alarm history", e);
            }
```

```
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error completing alarm history
processing", exception);
            }
        });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeAlarmHistory中的。

`DescribeAlarms`

以下代码示例演示了如何使用 `DescribeAlarms`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Describes the CloudWatch alarms of the 'METRIC_ALARM' type.
     *
     * @return a {@link CompletableFuture} that represents the asynchronous
operation
     * of describing the CloudWatch alarms. The future completes when the
     * operation is finished, either successfully or with an error.
     */
    public CompletableFuture<Void> describeAlarmsAsync() {
        List<AlarmType> typeList = new ArrayList<>();
        typeList.add(AlarmType.METRIC_ALARM);
        DescribeAlarmsRequest alarmsRequest = DescribeAlarmsRequest.builder()
            .alarmTypes(typeList)
            .maxRecords(10)
            .build();

        return getAsyncClient().describeAlarms(alarmsRequest)
```

```
            .thenAccept(response -> {
                List<MetricAlarm> alarmList = response.metricAlarms();
                for (MetricAlarm alarm : alarmList) {
                    logger.info("Alarm name: {}", alarm.alarmName());
                    logger.info("Alarm description: {} ", alarm.alarmDescription());
                }
            })
            .whenComplete((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to describe alarms: {}", ex.getMessage());
                } else {
                    logger.info("Successfully described alarms.");
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[DescribeAlarms](#)中的。

`DescribeAlarmsForMetric`

以下代码示例演示了如何使用 `DescribeAlarmsForMetric`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Checks for a metric alarm in AWS CloudWatch.
     *
     * @param fileName the name of the file containing the JSON configuration for
the custom metric
     * @return a {@link CompletableFuture} that completes when the check for the
metric alarm is complete
     */
    public CompletableFuture<Void> checkForMetricAlarmAsync(String fileName) {
```

```
        CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
                return rootNode.toString(); // Return JSON as a string for further
processing
            } catch (IOException e) {
                throw new RuntimeException("Failed to read file", e);
            }
        });

        return readFileFuture.thenCompose(jsonContent -> {
            try {
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();

                DescribeAlarmsForMetricRequest metricRequest =
DescribeAlarmsForMetricRequest.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

                return checkForAlarmAsync(metricRequest, customMetricName, 10);

            } catch (IOException e) {
                throw new RuntimeException("Failed to parse JSON content", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error checking metric alarm",
exception);
            }
        });
    }

    // Recursive method to check for the alarm.
```

```
   /**
    * Checks for the existence of an alarm asynchronously for the specified metric.
    *
    * @param metricRequest    the request to describe the alarms for the specified
metric
    * @param customMetricName the name of the custom metric to check for an alarm
    * @param retries          the number of retries to perform if no alarm is found
    * @return a {@link CompletableFuture} that completes when an alarm is found or
the maximum number of retries has been reached
    */
   private static CompletableFuture<Void>
checkForAlarmAsync(DescribeAlarmsForMetricRequest metricRequest, String
customMetricName, int retries) {
       if (retries == 0) {
           return CompletableFuture.completedFuture(null).thenRun(() ->
               logger.info("No Alarm state found for {} after 10 retries.",
customMetricName)
           );
       }

       return
(getAsyncClient().describeAlarmsForMetric(metricRequest).thenCompose(response -> {
           if (response.hasMetricAlarms()) {
               logger.info("Alarm state found for {}", customMetricName);
               return CompletableFuture.completedFuture(null); // Alarm found,
complete the future
           } else {
               return CompletableFuture.runAsync(() -> {
                   try {
                       Thread.sleep(20000);
                       logger.info(".");
                   } catch (InterruptedException e) {
                       throw new RuntimeException("Interrupted while waiting to
retry", e);
                   }
               }).thenCompose(v -> checkForAlarmAsync(metricRequest,
customMetricName, retries - 1)); // Recursive call
           }
       }));
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[DescribeAlarmsForMetric](#)中的。

DescribeAnomalyDetectors

以下代码示例演示了如何使用 DescribeAnomalyDetectors。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Describes the anomaly detectors based on the specified JSON file.
    *
    * @param fileName the name of the JSON file containing the custom metric
namespace and name
    * @return a {@link CompletableFuture} that completes when the anomaly detectors
have been described
    * @throws RuntimeException if there is a failure during the operation, such as
when reading or parsing the JSON file,
    *                          or when describing the anomaly detectors
    */
   public CompletableFuture<Void> describeAnomalyDetectorsAsync(String fileName) {
       CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               return new ObjectMapper().readTree(parser);
           } catch (IOException e) {
               throw new RuntimeException("Failed to read or parse the file", e);
           }
       });

       return readFileFuture.thenCompose(rootNode -> {
           try {
               String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
               String customMetricName =
rootNode.findValue("customMetricName").asText();
```

```
                DescribeAnomalyDetectorsRequest detectorsRequest =
DescribeAnomalyDetectorsRequest.builder()
                    .maxResults(10)
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

                return
getAsyncClient().describeAnomalyDetectors(detectorsRequest).thenAccept(response ->
{
                    List<AnomalyDetector> anomalyDetectorList =
response.anomalyDetectors();
                    for (AnomalyDetector detector : anomalyDetectorList) {
                        logger.info("Metric name: {} ",
detector.singleMetricAnomalyDetector().metricName());
                        logger.info("State: {} ", detector.stateValue());
                    }
                });
            } catch (RuntimeException e) {
                throw new RuntimeException("Failed to describe anomaly detectors",
e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error describing anomaly detectors",
exception);
            }
        });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[DescribeAnomalyDetectors](#)中的。

DisableAlarmActions

以下代码示例演示了如何使用 DisableAlarmActions。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DisableAlarmActions中的。

`EnableAlarmActions`

以下代码示例演示了如何使用 `EnableAlarmActions`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考EnableAlarmActions中的。

GetMetricData

以下代码示例演示了如何使用 GetMetricData。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Retrieves custom metric data from the AWS CloudWatch service.
    *
    * @param fileName the name of the file containing the custom metric information
    * @return a {@link CompletableFuture} that completes when the metric data has
been retrieved
    */
   public CompletableFuture<Void> getCustomMetricDataAsync(String fileName) {
       CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
           try {
               // Read values from the JSON file.
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
               return rootNode.toString(); // Return JSON as a string for further
processing
           } catch (IOException e) {
```

```
                throw new RuntimeException("Failed to read file", e);
            }
        });

        return readFileFuture.thenCompose(jsonContent -> {
            try {
                // Parse the JSON string to extract relevant values.
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();

                // Set the current time and date range for metric query.
                Instant nowDate = Instant.now();
                long hours = 1;
                long minutes = 30;
                Instant endTime = nowDate.plus(hours,
ChronoUnit.HOURS).plus(minutes, ChronoUnit.MINUTES);

                Metric met = Metric.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .build();

                MetricStat metStat = MetricStat.builder()
                    .stat("Maximum")
                    .period(60)  // Assuming period in seconds
                    .metric(met)
                    .build();

                MetricDataQuery dataQuery = MetricDataQuery.builder()
                    .metricStat(metStat)
                    .id("foo2")
                    .returnData(true)
                    .build();

                List<MetricDataQuery> dq = new ArrayList<>();
                dq.add(dataQuery);

                GetMetricDataRequest getMetricDataRequest =
GetMetricDataRequest.builder()
                    .maxDatapoints(10)
```

```
                    .scanBy(ScanBy.TIMESTAMP_DESCENDING)
                    .startTime(nowDate)
                    .endTime(endTime)
                    .metricDataQueries(dq)
                    .build();

                // Call the async method for CloudWatch data retrieval.
                return getAsyncClient().getMetricData(getMetricDataRequest);

            } catch (IOException e) {
                throw new RuntimeException("Failed to parse JSON content", e);
            }
        }).thenAccept(response -> {
            List<MetricDataResult> data = response.metricDataResults();
            for (MetricDataResult item : data) {
                logger.info("The label is: {}", item.label());
                logger.info("The status code is: {}", item.statusCode().toString());
            }
        }).exceptionally(exception -> {
            throw new RuntimeException("Failed to get metric data", exception);
        });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetMetricData中的。

`GetMetricStatistics`

以下代码示例演示了如何使用 `GetMetricStatistics`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Retrieves and displays metric statistics for the specified parameters.
     *
     * @param nameSpace    the namespace for the metric
```

```
    * @param metVal       the name of the metric
    * @param metricOption the statistic to retrieve for the metric (e.g.,
"Maximum", "Average")
    * @param date         the date for which to retrieve the metric statistics, in
the format "yyyy-MM-dd'T'HH:mm:ss'Z'"
    * @param myDimension  the dimension(s) to filter the metric statistics by
    * @return a {@link CompletableFuture} that completes when the metric statistics
have been retrieved and displayed
    */
   public CompletableFuture<GetMetricStatisticsResponse>
getAndDisplayMetricStatisticsAsync(String nameSpace, String metVal,

        String metricOption, String date, Dimension myDimension) {

       Instant start = Instant.parse(date);
       Instant endDate = Instant.now();

       // Building the request for metric statistics.
       GetMetricStatisticsRequest statisticsRequest =
GetMetricStatisticsRequest.builder()
           .endTime(endDate)
           .startTime(start)
           .dimensions(myDimension)
           .metricName(metVal)
           .namespace(nameSpace)
           .period(86400) // 1 day period
           .statistics(Statistic.fromValue(metricOption))
           .build();

       return getAsyncClient().getMetricStatistics(statisticsRequest)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   List<Datapoint> data = response.datapoints();
                   if (!data.isEmpty()) {
                       for (Datapoint datapoint : data) {
                           logger.info("Timestamp: {} Maximum value: {}",
datapoint.timestamp(), datapoint.maximum());
                       }
                   } else {
                       logger.info("The returned data list is empty");
                   }
               } else {
                   logger.info("Failed to get metric statistics: {} ",
exception.getMessage());
```

```
                }
            })
            .exceptionally(exception -> {
                throw new RuntimeException("Error while getting metric statistics: "
+ exception.getMessage(), exception);
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetMetricStatistics中的。

`GetMetricWidgetImage`

以下代码示例演示了如何使用 `GetMetricWidgetImage`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Retrieves and saves a custom metric image to a file.
     *
     * @param fileName the name of the file to save the metric image to
     * @return a {@link CompletableFuture} that completes when the image has been
saved to the file
     */
    public CompletableFuture<Void> downloadAndSaveMetricImageAsync(String fileName)
{
        logger.info("Getting Image data for custom metric.");
        String myJSON = """
              {
                  "title": "Example Metric Graph",
                  "view": "timeSeries",
                  "stacked ": false,
                  "period": 10,
                  "width": 1400,
```

```
                "height": 600,
                "metrics": [
                    [
                    "AWS/Billing",
                    "EstimatedCharges",
                    "Currency",
                    "USD"
                   ]
                ]
              }
            """;

       GetMetricWidgetImageRequest imageRequest =
GetMetricWidgetImageRequest.builder()
            .metricWidget(myJSON)
            .build();

       return getAsyncClient().getMetricWidgetImage(imageRequest)
            .thenCompose(response -> {
                SdkBytes sdkBytes = response.metricWidgetImage();
                byte[] bytes = sdkBytes.asByteArray();
                return CompletableFuture.runAsync(() -> {
                    try {
                        File outputFile = new File(fileName);
                        try (FileOutputStream outputStream = new
FileOutputStream(outputFile)) {
                            outputStream.write(bytes);
                        }
                    } catch (IOException e) {
                        throw new RuntimeException("Failed to write image to file",
e);
                    }
                });
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Error getting and saving metric
image", exception);
                } else {
                    logger.info("Image data saved successfully to {}", fileName);
                }
            });
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetMetricWidgetImage中的。

`ListDashboards`

以下代码示例演示了如何使用 `ListDashboards`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Lists the available dashboards.
    *
    * @return a {@link CompletableFuture} that completes when the operation is
finished.
    * The future will complete exceptionally if an error occurs while listing the
dashboards.
    */
   public CompletableFuture<Void> listDashboardsAsync() {
       ListDashboardsRequest listDashboardsRequest =
ListDashboardsRequest.builder().build();
       ListDashboardsPublisher paginator =
getAsyncClient().listDashboardsPaginator(listDashboardsRequest);
       return paginator.subscribe(response -> {
           response.dashboardEntries().forEach(entry -> {
               logger.info("Dashboard name is: {} ", entry.dashboardName());
               logger.info("Dashboard ARN is: {} ", entry.dashboardArn());
           });
       }).exceptionally(ex -> {
           logger.info("Failed to list dashboards: {} ", ex.getMessage());
           throw new RuntimeException("Error occurred while listing dashboards",
ex);
       });
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListDashboards中的。

`ListMetrics`

以下代码示例演示了如何使用 `ListMetrics`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Retrieves a list of metric names for the specified namespace.
     *
     * @param namespace the namespace for which to retrieve the metric names
     * @return a {@link CompletableFuture} that, when completed, contains an {@link
ArrayList} of
     * the metric names in the specified namespace
     * @throws RuntimeException if an error occurs while listing the metrics
     */
    public CompletableFuture<ArrayList<String>> listMetsAsync(String namespace) {
        ListMetricsRequest request = ListMetricsRequest.builder()
            .namespace(namespace)
            .build();

        ListMetricsPublisher metricsPaginator =
getAsyncClient().listMetricsPaginator(request);
        Set<String> metSet = new HashSet<>();
        CompletableFuture<Void> future = metricsPaginator.subscribe(response -> {
            response.metrics().forEach(metric -> {
                String metricName = metric.metricName();
                metSet.add(metricName);
            });
        });

        return future
            .thenApply(ignored -> new ArrayList<>(metSet))
            .exceptionally(exception -> {
```

```
                throw new RuntimeException("Failed to list metrics: " +
exception.getMessage(), exception);
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListMetrics中的。

`PutAnomalyDetector`

以下代码示例演示了如何使用 `PutAnomalyDetector`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Adds an anomaly detector for the given file.
     *
     * @param fileName the name of the file containing the anomaly detector
configuration
     * @return a {@link CompletableFuture} that completes when the anomaly detector
has been added
     */
    public CompletableFuture<Void> addAnomalyDetectorAsync(String fileName) {
        CompletableFuture<JsonNode> readFileFuture =
CompletableFuture.supplyAsync(() -> {
            try {
                JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                return new ObjectMapper().readTree(parser); // Return the root node
            } catch (IOException e) {
                throw new RuntimeException("Failed to read or parse the file", e);
            }
        });
```

```
        return readFileFuture.thenCompose(rootNode -> {
            try {
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();

                SingleMetricAnomalyDetector singleMetricAnomalyDetector =
SingleMetricAnomalyDetector.builder()
                    .metricName(customMetricName)
                    .namespace(customMetricNamespace)
                    .stat("Maximum")
                    .build();

                PutAnomalyDetectorRequest anomalyDetectorRequest =
PutAnomalyDetectorRequest.builder()
                    .singleMetricAnomalyDetector(singleMetricAnomalyDetector)
                    .build();

                return
getAsyncClient().putAnomalyDetector(anomalyDetectorRequest).thenAccept(response ->
{
                    logger.info("Added anomaly detector for metric {}",
customMetricName);
                });
            } catch (Exception e) {
                throw new RuntimeException("Failed to create anomaly detector", e);
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                throw new RuntimeException("Error adding anomaly detector",
exception);
            }
        });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[PutAnomalyDetector](#)中的。

PutDashboard

以下代码示例演示了如何使用 PutDashboard。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Creates a new dashboard with the specified name and metrics from the given
file.
    *
    * @param dashboardName the name of the dashboard to be created
    * @param fileName      the name of the file containing the dashboard body
    * @return a {@link CompletableFuture} representing the asynchronous operation
of creating the dashboard
    * @throws IOException if there is an error reading the dashboard body from the
file
    */
   public CompletableFuture<PutDashboardResponse>
createDashboardWithMetricsAsync(String dashboardName, String fileName) throws
IOException {
       String dashboardBody = readFileAsString(fileName);
       PutDashboardRequest dashboardRequest = PutDashboardRequest.builder()
           .dashboardName(dashboardName)
           .dashboardBody(dashboardBody)
           .build();

       return getAsyncClient().putDashboard(dashboardRequest)
           .handle((response, ex) -> {
               if (ex != null) {
                   logger.info("Failed to create dashboard: {}", ex.getMessage());
                   throw new RuntimeException("Dashboard creation failed", ex);
               } else {
                   // Handle the normal response case
                   logger.info("{} was successfully created.", dashboardName);
                   List<DashboardValidationMessage> messages =
response.dashboardValidationMessages();
                   if (messages.isEmpty()) {
                       logger.info("There are no messages in the new Dashboard.");
                   } else {
```

```
                    for (DashboardValidationMessage message : messages) {
                        logger.info("Message: {}", message.message());
                    }
                }
                return response; // Return the response for further use
            }
        });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutDashboard中的。

`PutMetricAlarm`

以下代码示例演示了如何使用 `PutMetricAlarm`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Creates an alarm based on the configuration provided in a JSON file.
     *
     * @param fileName the name of the JSON file containing the alarm configuration
     * @return a CompletableFuture that represents the asynchronous operation of
creating the alarm
     * @throws RuntimeException if an exception occurs while reading the JSON file
or creating the alarm
     */
    public CompletableFuture<String> createAlarmAsync(String fileName) {
        com.fasterxml.jackson.databind.JsonNode rootNode;
        try {
            JsonParser parser = new JsonFactory().createParser(new File(fileName));
            rootNode = new ObjectMapper().readTree(parser);
        } catch (IOException e) {
            throw new RuntimeException("Failed to read the alarm configuration
file", e);
        }
```

```
        // Extract values from the JSON node.
        String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
        String customMetricName = rootNode.findValue("customMetricName").asText();
        String alarmName = rootNode.findValue("exampleAlarmName").asText();
        String emailTopic = rootNode.findValue("emailTopic").asText();
        String accountId = rootNode.findValue("accountId").asText();
        String region = rootNode.findValue("region").asText();

        // Create a List for alarm actions.
        List<String> alarmActions = new ArrayList<>();
        alarmActions.add("arn:aws:sns:" + region + ":" + accountId + ":" +
emailTopic);

        PutMetricAlarmRequest alarmRequest = PutMetricAlarmRequest.builder()
            .alarmActions(alarmActions)
            .alarmDescription("Example metric alarm")
            .alarmName(alarmName)

.comparisonOperator(ComparisonOperator.GREATER_THAN_OR_EQUAL_TO_THRESHOLD)
            .threshold(100.00)
            .metricName(customMetricName)
            .namespace(customMetricNamespace)
            .evaluationPeriods(1)
            .period(10)
            .statistic("Maximum")
            .datapointsToAlarm(1)
            .treatMissingData("ignore")
            .build();

        // Call the putMetricAlarm asynchronously and handle the result.
        return getAsyncClient().putMetricAlarm(alarmRequest)
            .handle((response, ex) -> {
                if (ex != null) {
                    logger.info("Failed to create alarm: {}", ex.getMessage());
                    throw new RuntimeException("Failed to create alarm", ex);
                } else {
                    logger.info("{} was successfully created!", alarmName);
                    return alarmName;
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutMetricAlarm中的。

`PutMetricData`

以下代码示例演示了如何使用 `PutMetricData`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Adds metric data for an alarm asynchronously.
    *
    * @param fileName the name of the JSON file containing the metric data
    * @return a CompletableFuture that asynchronously returns the
PutMetricDataResponse
    */
   public CompletableFuture<PutMetricDataResponse>
addMetricDataForAlarmAsync(String fileName) {
       CompletableFuture<String> readFileFuture = CompletableFuture.supplyAsync(()
-> {
           try {
               JsonParser parser = new JsonFactory().createParser(new
File(fileName));
               com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
               return rootNode.toString(); // Return JSON as a string for further
processing
           } catch (IOException e) {
               throw new RuntimeException("Failed to read file", e);
           }
       });

       return readFileFuture.thenCompose(jsonContent -> {
           try {
```

```
                com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(jsonContent);
                String customMetricNamespace =
rootNode.findValue("customMetricNamespace").asText();
                String customMetricName =
rootNode.findValue("customMetricName").asText();
                Instant instant = Instant.now();

                // Create MetricDatum objects.
                MetricDatum datum1 = MetricDatum.builder()
                    .metricName(customMetricName)
                    .unit(StandardUnit.NONE)
                    .value(1001.00)
                    .timestamp(instant)
                    .build();

                MetricDatum datum2 = MetricDatum.builder()
                    .metricName(customMetricName)
                    .unit(StandardUnit.NONE)
                    .value(1002.00)
                    .timestamp(instant)
                    .build();

                List<MetricDatum> metricDataList = new ArrayList<>();
                metricDataList.add(datum1);
                metricDataList.add(datum2);

                // Build the PutMetricData request.
                PutMetricDataRequest request = PutMetricDataRequest.builder()
                    .namespace(customMetricNamespace)
                    .metricData(metricDataList)
                    .build();

                // Send the request asynchronously.
                return getAsyncClient().putMetricData(request);

            } catch (IOException e) {
                CompletableFuture<PutMetricDataResponse> failedFuture = new
CompletableFuture<>();
                failedFuture.completeExceptionally(new RuntimeException("Failed to
parse JSON content", e));
                return failedFuture;
            }
        }).whenComplete((response, exception) -> {
```

```
            if (exception != null) {
                logger.error("Failed to put metric data: " + exception.getMessage(),
exception);
            } else {
                logger.info("Added metric values for metric.");
            }
        });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutMetricData中的。

场景

监控 DynamoDB 性能

以下代码示例显示如何配置应用程序使用 DynamoDB 来监控性能。

适用于 Java 的 SDK 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

CloudWatch 使用适用于 Java 的 SDK 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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从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

以下代码示例演示了如何使用 PutEvents。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutEvents中的。

PutRule

以下代码示例演示了如何使用 PutRule。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[PutRule](#)中的。

PutTargets

以下代码示例演示了如何使用 PutTargets。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutTargets中的。

CloudWatch 使用适用于 Java 的 SDK 2.x 记录示例

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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从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

`DeleteSubscriptionFilter`

以下代码示例演示了如何使用 `DeleteSubscriptionFilter`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteSubscriptionFilter中的。

`DescribeLogStreams`

以下代码示例演示了如何使用 `DescribeLogStreams`。

适用于 Java 的 SDK 2.x

> **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。

在指定日志组中搜索与给定前缀匹配的日志流。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CloudWatchLogsSearch {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                  <logGroupName> <logStreamName>

                Where:
                  logGroupName - The name of the log group (for example,
 WeathertopJavaContainerLogs).
                  logStreamName - The name of the log stream (for example,
 weathertop-java-stream).
```

```
                  pattern - the pattern to use (for example, INFO)

                """;

        if (args.length != 3) {
            System.out.print(usage);
            System.exit(1);
        }

        String logGroupName = args[0] ;
        String logStreamName = args[1] ;
        String pattern = args[2] ;

        CloudWatchLogsClient cwlClient = CloudWatchLogsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        searchLogStreamsAndFilterEvents(cwlClient, logGroupName, logStreamName,
pattern);
    }

    /**
     * Searches for log streams with a specific prefix within a log group and
filters log events based on a specified pattern.
     *
     * @param cwlClient       the CloudWatchLogsClient used to interact with AWS
CloudWatch Logs
     * @param logGroupName    the name of the log group to search within
     * @param logStreamPrefix the prefix of the log streams to search for
     * @param pattern         the pattern to filter log events by
     */
    public static void searchLogStreamsAndFilterEvents(CloudWatchLogsClient
cwlClient, String logGroupName, String logStreamPrefix, String pattern) {
        DescribeLogStreamsRequest describeLogStreamsRequest =
DescribeLogStreamsRequest.builder()
                .logGroupName(logGroupName)
                .logStreamNamePrefix(logStreamPrefix)
                .build();

        DescribeLogStreamsResponse describeLogStreamsResponse =
cwlClient.describeLogStreams(describeLogStreamsRequest);
        List<LogStream> logStreams = describeLogStreamsResponse.logStreams();

        for (LogStream logStream : logStreams) {
```

```
            String logStreamName = logStream.logStreamName();
            System.out.println("Searching in log stream: " + logStreamName);

            FilterLogEventsRequest filterLogEventsRequest =
 FilterLogEventsRequest.builder()
                    .logGroupName(logGroupName)
                    .logStreamNames(logStreamName)
                    .filterPattern(pattern)
                    .build();

            FilterLogEventsResponse filterLogEventsResponse =
 cwlClient.filterLogEvents(filterLogEventsRequest);

            for (FilteredLogEvent event : filterLogEventsResponse.events()) {
                System.out.println(event.message());
            }


 System.out.println("-------------------------------------------------"); //
 Separator for better readability
        }
    }
}
```

打印有关指定日志组中最新日志流的元数据。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CloudWatchLogQuery {
    public static void main(final String[] args) {
        final String usage = """
                Usage:
                  <logGroupName>

                Where:
```

```
                  logGroupName - The name of the log group (for example, /aws/
lambda/ChatAIHandler).
                """";

        if (args.length != 1) {
            System.out.print(usage);
            System.exit(1);
        }

        String logGroupName = "/aws/lambda/ChatAIHandler" ; //args[0];
        CloudWatchLogsClient logsClient = CloudWatchLogsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        describeMostRecentLogStream(logsClient, logGroupName);
    }

    /**
     * Describes and prints metadata about the most recent log stream in the
 specified log group.
     *
     * @param logsClient   the CloudWatchLogsClient used to interact with AWS
 CloudWatch Logs
     * @param logGroupName the name of the log group
     */
    public static void describeMostRecentLogStream(CloudWatchLogsClient logsClient,
 String logGroupName) {
        DescribeLogStreamsRequest streamsRequest =
 DescribeLogStreamsRequest.builder()
                .logGroupName(logGroupName)
                .orderBy(OrderBy.LAST_EVENT_TIME)
                .descending(true)
                .limit(1)
                .build();

        try {
            DescribeLogStreamsResponse streamsResponse =
 logsClient.describeLogStreams(streamsRequest);
            List<LogStream> logStreams = streamsResponse.logStreams();

            if (logStreams.isEmpty()) {
                System.out.println("No log streams found for log group: " +
 logGroupName);
                return;
```

```
            }

            LogStream stream = logStreams.get(0);
            System.out.println("Most Recent Log Stream:");
            System.out.println("  Name: " + stream.logStreamName());
            System.out.println("  ARN: " + stream.arn());
            System.out.println("  Creation Time: " + stream.creationTime());
            System.out.println("  First Event Time: " +
 stream.firstEventTimestamp());
            System.out.println("  Last Event Time: " + stream.lastEventTimestamp());
            System.out.println("  Stored Bytes: " + stream.storedBytes());
            System.out.println("  Upload Sequence Token: " +
 stream.uploadSequenceToken());

        } catch (CloudWatchLogsException e) {
            System.err.println("Failed to describe log stream: " +
 e.awsErrorDetails().errorMessage());
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeLogStreams中的。

`DescribeSubscriptionFilters`

以下代码示例演示了如何使用 `DescribeSubscriptionFilters`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[DescribeSubscriptionFilters](#)中的。

GetLogEvents

以下代码示例演示了如何使用 GetLogEvents。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.model.CloudWatchException;
import software.amazon.awssdk.services.cloudwatchlogs.CloudWatchLogsClient;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DescribeLogStreamsRequest;
import
 software.amazon.awssdk.services.cloudwatchlogs.model.DescribeLogStreamsResponse;
import software.amazon.awssdk.services.cloudwatchlogs.model.GetLogEventsRequest;
import software.amazon.awssdk.services.cloudwatchlogs.model.GetLogEventsResponse;

import java.time.Instant;
import java.time.temporal.ChronoUnit;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetLogEvents {

    public static void main(String[] args) {

        final String usage = """

                Usage:
                  <logGroupName> <logStreamName>

                Where:
```

```
                    logGroupName - The name of the log group (for example,
 myloggroup).
                    logStreamName - The name of the log stream (for example,
 mystream).

                """;

        // if (args.length != 2) {
        //     System.out.print(usage);
        //     System.exit(1);
//        }

         String logGroupName = "WeathertopJavaContainerLogs" ; //args[0];
         String logStreamName = "weathertop-java-stream" ; //args[1];

         Region region = Region.US_EAST_1 ;
         CloudWatchLogsClient cloudWatchLogsClient = CloudWatchLogsClient.builder()
                 .region(region)
                 .build();

         getCWLogEvents(cloudWatchLogsClient, logGroupName, logStreamName);
         cloudWatchLogsClient.close();
    }

    public static void getCWLogEvents(CloudWatchLogsClient cloudWatchLogsClient,
                                      String logGroupName,
                                      String logStreamPrefix) {
         try {
             // First, find the exact log stream name
             DescribeLogStreamsRequest describeRequest =
 DescribeLogStreamsRequest.builder()
                     .logGroupName(logGroupName)
                     .logStreamNamePrefix(logStreamPrefix)
                     .limit(1) // get the first matching stream
                     .build();

             DescribeLogStreamsResponse describeResponse =
 cloudWatchLogsClient.describeLogStreams(describeRequest);

             if (describeResponse.logStreams().isEmpty()) {
                 System.out.println("No matching log streams found for prefix: " +
 logStreamPrefix);
                 return;
             }
```

```
            String exactLogStreamName =
 describeResponse.logStreams().get(0).logStreamName();
            System.out.println("Using exact log stream: " + exactLogStreamName);

            long startTime = Instant.now().minus(7, ChronoUnit.DAYS).toEpochMilli();
            long endTime = Instant.now().toEpochMilli();

            GetLogEventsRequest getLogEventsRequest = GetLogEventsRequest.builder()
                    .logGroupName(logGroupName)
                    .logStreamName(exactLogStreamName) // <-- exact name, not prefix
                    .startTime(startTime)
                    .endTime(endTime)
                    .startFromHead(true)
                    .build();

            GetLogEventsResponse response =
 cloudWatchLogsClient.getLogEvents(getLogEventsRequest);

            if (response.events().isEmpty()) {
                System.out.println("No log events found in the past 7 days.");
            } else {
                response.events().forEach(e -> System.out.println(e.message()));
            }

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

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[GetLogEvents](#)中的。

PutSubscriptionFilter

以下代码示例演示了如何使用 PutSubscriptionFilter。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutSubscriptionFilter中的。

`StartLiveTail`

以下代码示例演示了如何使用 `StartLiveTail`。

适用于 Java 的 SDK 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartLiveTail中的。

场景

使用计划的事件调用 Lambda 函数

以下代码示例显示如何创建由 Amazon EventBridge 计划事件调用的 AWS Lambda 函数。

适用于 Java 的 SDK 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。在本示例中，您使用 Lambda Java 运行时 API 创建 Lambda 函数。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

- CloudWatch 日志
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

使用 SDK for Java 2.x 的 Amazon Cognito Identity 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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从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`

以下代码示例演示了如何使用 `CreateIdentityPool`。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateIdentityPool {
    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <identityPoolName>\s

                Where:
                    identityPoolName - The name to give your identity pool.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityPoolName = args[0];
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String identityPoolId = createIdPool(cognitoClient, identityPoolName);
        System.out.println("Unity pool ID " + identityPoolId);
        cognitoClient.close();
    }

    public static String createIdPool(CognitoIdentityClient cognitoClient, String
 identityPoolName) {
        try {
            CreateIdentityPoolRequest poolRequest =
 CreateIdentityPoolRequest.builder()
                    .allowUnauthenticatedIdentities(false)
                    .identityPoolName(identityPoolName)
                    .build();

            CreateIdentityPoolResponse response =
 cognitoClient.createIdentityPool(poolRequest);
            return response.identityPoolId();

        } catch (CognitoIdentityProviderException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateIdentityPool中的。

`DeleteIdentityPool`

以下代码示例演示了如何使用 `DeleteIdentityPool`。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.DeleteIdentityPoolRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteIdentityPool {

    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:
                    <identityPoolId>\s

                Where:
                    identityPoolId - The Id value of your identity pool.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityPoold = args[0];
        CognitoIdentityClient cognitoIdClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();

        deleteIdPool(cognitoIdClient, identityPoold);
        cognitoIdClient.close();
    }

    public static void deleteIdPool(CognitoIdentityClient cognitoIdClient, String
 identityPoold) {
        try {

            DeleteIdentityPoolRequest identityPoolRequest =
 DeleteIdentityPoolRequest.builder()
                    .identityPoolId(identityPoold)
                    .build();

            cognitoIdClient.deleteIdentityPool(identityPoolRequest);
            System.out.println("Done");

        } catch (AwsServiceException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[DeleteIdentityPool](#)中的。

GetCredentialsForIdentity

以下代码示例演示了如何使用 GetCredentialsForIdentity。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetIdentityCredentials {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <identityId>\s

            Where:
                identityId - The Id of an existing identity in the format
 REGION:GUID.
            """;

        if (args.length != 1) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String identityId = args[0];
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        getCredsForIdentity(cognitoClient, identityId);
        cognitoClient.close();
    }

    public static void getCredsForIdentity(CognitoIdentityClient cognitoClient,
 String identityId) {
        try {
            GetCredentialsForIdentityRequest getCredentialsForIdentityRequest =
 GetCredentialsForIdentityRequest
                    .builder()
                    .identityId(identityId)
                    .build();

            GetCredentialsForIdentityResponse response = cognitoClient
                    .getCredentialsForIdentity(getCredentialsForIdentityRequest);
            System.out.println(
                    "Identity ID " + response.identityId() + ", Access key ID " +
 response.credentials().accessKeyId());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetCredentialsForIdentity中的。

ListIdentityPools

以下代码示例演示了如何使用 ListIdentityPools。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListIdentityPools {
    public static void main(String[] args) {
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listIdPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listIdPools(CognitoIdentityClient cognitoClient) {
        try {
            ListIdentityPoolsRequest poolsRequest =
 ListIdentityPoolsRequest.builder()
                    .maxResults(15)
                    .build();
```

```
            ListIdentityPoolsResponse response =
 cognitoClient.listIdentityPools(poolsRequest);
            response.identityPools().forEach(pool -> {
                System.out.println("Pool ID: " + pool.identityPoolId());
                System.out.println("Pool name: " + pool.identityPoolName());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListIdentityPools中的。

使用 SDK for Java 2.x 的 Amazon Cognito 身份提供者示例

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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 操作
- 场景

开始使用

开始使用 Amazon Cognito

以下代码示例显示如何开始使用 Amazon Cognito。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ListUserPools](#)中的。

操作

`AdminGetUser`

以下代码示例演示了如何使用 `AdminGetUser`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考AdminGetUser中的。

`AdminInitiateAuth`

以下代码示例演示了如何使用 `AdminInitiateAuth`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考AdminInitiateAuth中的。

`AdminRespondToAuthChallenge`

以下代码示例演示了如何使用 `AdminRespondToAuthChallenge`。

适用于 Java 的 SDK 2.x

> **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。

```
    // 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[AdminRespondToAuthChallenge](#)中的。

`AssociateSoftwareToken`

以下代码示例演示了如何使用 `AssociateSoftwareToken`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[AssociateSoftwareToken](#)中的。

`ConfirmSignUp`

以下代码示例演示了如何使用 `ConfirmSignUp`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ConfirmSignUp中的。

`CreateUserPool`

以下代码示例演示了如何使用 `CreateUserPool`。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateUserPool {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <userPoolName>\s

                Where:
```

```
                    userPoolName - The name to give your user pool when it's
 created.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userPoolName = args[0];
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String id = createPool(cognitoClient, userPoolName);
        System.out.println("User pool ID: " + id);
        cognitoClient.close();
    }

    public static String createPool(CognitoIdentityProviderClient cognitoClient,
 String userPoolName) {
        try {
            CreateUserPoolRequest request = CreateUserPoolRequest.builder()
                    .poolName(userPoolName)
                    .build();

            CreateUserPoolResponse response = cognitoClient.createUserPool(request);
            return response.userPool().id();

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateUserPool中的。

CreateUserPoolClient

以下代码示例演示了如何使用 CreateUserPoolClient。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientResponse;

/**
 * A user pool client app is an application that authenticates with Amazon
 * Cognito user pools.
 * When you create a user pool, you can configure app clients that allow mobile
 * or web applications
 * to call API operations to authenticate users, manage user attributes and
 * profiles,
 * and implement sign-up and sign-in flows.
 *
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateUserPoolClient {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <clientName> <userPoolId>\s
```

```
                Where:
                    clientName - The name for the user pool client to create.
                    userPoolId - The ID for the user pool.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clientName = args[0];
        String userPoolId = args[1];
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createPoolClient(cognitoClient, clientName, userPoolId);
        cognitoClient.close();
    }

    public static void createPoolClient(CognitoIdentityProviderClient cognitoClient,
 String clientName,
            String userPoolId) {
        try {
            CreateUserPoolClientRequest request =
 CreateUserPoolClientRequest.builder()
                    .clientName(clientName)
                    .userPoolId(userPoolId)
                    .build();

            CreateUserPoolClientResponse response =
 cognitoClient.createUserPoolClient(request);
            System.out.println("User pool " + response.userPoolClient().clientName()
 + " created. ID: "
                    + response.userPoolClient().clientId());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateUserPoolClient中的。

`ListUserPools`

以下代码示例演示了如何使用 `ListUserPools`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListUserPools中的。

`ListUsers`

以下代码示例演示了如何使用 `ListUsers`。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
```

```
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListUsers {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <userPoolId>\s

                Where:
                    userPoolId - The ID given to your user pool when it's created.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userPoolId = args[0];
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllUsers(cognitoClient, userPoolId);
        listUsersFilter(cognitoClient, userPoolId);
        cognitoClient.close();
    }
```

```
    public static void listAllUsers(CognitoIdentityProviderClient cognitoClient,
 String userPoolId) {
        try {
            ListUsersRequest usersRequest = ListUsersRequest.builder()
                    .userPoolId(userPoolId)
                    .build();

            ListUsersResponse response = cognitoClient.listUsers(usersRequest);
            response.users().forEach(user -> {
                System.out.println("User " + user.username() + " Status " +
 user.userStatus() + " Created "
                        + user.userCreateDate());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Shows how to list users by using a filter.
    public static void listUsersFilter(CognitoIdentityProviderClient cognitoClient,
 String userPoolId) {

        try {
            String filter = "email = \"tblue@noserver.com\"";
            ListUsersRequest usersRequest = ListUsersRequest.builder()
                    .userPoolId(userPoolId)
                    .filter(filter)
                    .build();

            ListUsersResponse response = cognitoClient.listUsers(usersRequest);
            response.users().forEach(user -> {
                System.out.println("User with filter applied " + user.username() + "
 Status " + user.userStatus()
                        + " Created " + user.userCreateDate());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListUsers中的。

`ResendConfirmationCode`

以下代码示例演示了如何使用 `ResendConfirmationCode`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ResendConfirmationCode中的。

SignUp

以下代码示例演示了如何使用 SignUp。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SignUp中的。

`VerifySoftwareToken`

以下代码示例演示了如何使用 `VerifySoftwareToken`。

适用于 Java 的 SDK 2.x

> **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。

```
   // 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考VerifySoftwareToken中的。

场景

向需要 MFA 的用户池注册用户

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重身份验证。
- 使用密码和 MFA 代码登录。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChallengeRe
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChallengeRe
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenRequest
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenRespons
import software.amazon.awssdk.services.cognitoidentityprovider.model.AttributeType;
import software.amazon.awssdk.services.cognitoidentityprovider.model.AuthFlowType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ChallengeNameType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExcept
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ConfirmSignUpRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeRequest
```

```
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeRespons
import software.amazon.awssdk.services.cognitoidentityprovider.model.SignUpRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenResponse;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * TIP: To set up the required user pool, run the AWS Cloud Development Kit (AWS
 * CDK) script provided in this GitHub repo at
 * resources/cdk/cognito_scenario_user_pool_with_mfa.
 *
 * This code example performs the following operations:
 *
 * 1. Invokes the signUp method to sign up a user.
 * 2. Invokes the adminGetUser method to get the user's confirmation status.
 * 3. Invokes the ResendConfirmationCode method if the user requested another
 * code.
 * 4. Invokes the confirmSignUp method.
 * 5. Invokes the AdminInitiateAuth to sign in. This results in being prompted
 * to set up TOTP (time-based one-time password). (The response is
 * “ChallengeName”: “MFA_SETUP”).
 * 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private
 * key. This can be used with Google Authenticator.
 * 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for
 * MFA.
 * 8. Invokes the AdminInitiateAuth to sign in again. This results in being
 * prompted to submit a TOTP (Response: “ChallengeName”: “SOFTWARE_TOKEN_MFA”).
 * 9. Invokes the AdminRespondToAuthChallenge to get back a token.
```

```
 */

public class CognitoMVP {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws NoSuchAlgorithmException,
 InvalidKeyException {
        final String usage = """

                Usage:
                    <clientId> <poolId>

                Where:
                    clientId - The app client Id value that you can get from the AWS
 CDK script.
                    poolId - The pool Id that you can get from the AWS CDK script.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clientId = args[0];
        String poolId = args[1];
        CognitoIdentityProviderClient identityProviderClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Cognito example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter your user name");
        Scanner in = new Scanner(System.in);
        String userName = in.nextLine();

        System.out.println("*** Enter your password");
        String password = in.nextLine();

        System.out.println("*** Enter your email");
        String email = in.nextLine();
```

```
        System.out.println("1. Signing up " + userName);
        signUp(identityProviderClient, clientId, userName, password, email);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Getting " + userName + " in the user pool");
        getAdminUser(identityProviderClient, userName, poolId);

        System.out
                .println("*** Conformation code sent to " + userName + ". Would you
like to send a new code? (Yes/No)");
        System.out.println(DASHES);

        System.out.println(DASHES);
        String ans = in.nextLine();

        if (ans.compareTo("Yes") == 0) {
            resendConfirmationCode(identityProviderClient, clientId, userName);
            System.out.println("3. Sending a new confirmation code");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Enter confirmation code that was emailed");
        String code = in.nextLine();
        confirmSignUp(identityProviderClient, clientId, code, userName);
        System.out.println("Rechecking the status of " + userName + " in the user
pool");
        getAdminUser(identityProviderClient, userName, poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Invokes the initiateAuth to sign in");
        AdminInitiateAuthResponse authResponse =
initiateAuth(identityProviderClient, clientId, userName, password,
                poolId);
        String mySession = authResponse.session();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Invokes the AssociateSoftwareToken method to generate
a TOTP key");
        String newSession = getSecretForAppMFA(identityProviderClient, mySession);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter the 6-digit code displayed in Google
Authenticator");
        String myCode = in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Verify the TOTP and register for MFA");
        verifyTOTP(identityProviderClient, newSession, myCode);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Re-enter a 6-digit code displayed in Google
Authenticator");
        String mfaCode = in.nextLine();
        AdminInitiateAuthResponse authResponse1 =
initiateAuth(identityProviderClient, clientId, userName, password,
                poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9.  Invokes the AdminRespondToAuthChallenge");
        String session2 = authResponse1.session();
        adminRespondToAuthChallenge(identityProviderClient, userName, clientId,
mfaCode, session2);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("All Amazon Cognito operations were successfully
performed");
        System.out.println(DASHES);
    }

    // Respond to an authentication challenge.
    public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient,
            String userName, String clientId, String mfaCode, String session) {
        System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
        Map<String, String> challengeResponses = new HashMap<>();

        challengeResponses.put("USERNAME", userName);
        challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);
```

```
        AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
                .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
                .clientId(clientId)
                .challengeResponses(challengeResponses)
                .session(session)
                .build();

        AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
                .adminRespondToAuthChallenge(respondToAuthChallengeRequest);
        System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
                + respondToAuthChallengeResult.authenticationResult());
    }

    // Verify the TOTP and register for MFA.
    public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
        try {
            VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
                    .userCode(code)
                    .session(session)
                    .build();

            VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
            System.out.println("The status of the token is " +
verifyResponse.statusAsString());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static AdminInitiateAuthResponse
initiateAuth(CognitoIdentityProviderClient identityProviderClient,
            String clientId, String userName, String password, String userPoolId) {
        try {
            Map<String, String> authParameters = new HashMap<>();
            authParameters.put("USERNAME", userName);
            authParameters.put("PASSWORD", password);
```

```
            AdminInitiateAuthRequest authRequest =
AdminInitiateAuthRequest.builder()
                    .clientId(clientId)
                    .userPoolId(userPoolId)
                    .authParameters(authParameters)
                    .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
                    .build();

            AdminInitiateAuthResponse response =
identityProviderClient.adminInitiateAuth(authRequest);
            System.out.println("Result Challenge is : " + response.challengeName());
            return response;

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }

    public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
        AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
                .session(session)
                .build();

        AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
                .associateSoftwareToken(softwareTokenRequest);
        String secretCode = tokenResponse.secretCode();
        System.out.println("Enter this token into Google Authenticator");
        System.out.println(secretCode);
        return tokenResponse.session();
    }

    public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code,
            String userName) {
        try {
            ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
                    .clientId(clientId)
                    .confirmationCode(code)
```

```
                    .username(userName)
                    .build();

            identityProviderClient.confirmSignUp(signUpRequest);
            System.out.println(userName + " was confirmed");

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
            String userName) {
        try {
            ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
                    .clientId(clientId)
                    .username(userName)
                    .build();

            ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
            System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void signUp(CognitoIdentityProviderClient identityProviderClient,
String clientId, String userName,
            String password, String email) {
        AttributeType userAttrs = AttributeType.builder()
                .name("email")
                .value(email)
                .build();

        List<AttributeType> userAttrsList = new ArrayList<>();
        userAttrsList.add(userAttrs);
        try {
```

```
            SignUpRequest signUpRequest = SignUpRequest.builder()
                    .userAttributes(userAttrsList)
                    .username(userName)
                    .clientId(clientId)
                    .password(password)
                    .build();

            identityProviderClient.signUp(signUpRequest);
            System.out.println("User has been signed up ");

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getAdminUser(CognitoIdentityProviderClient
 identityProviderClient, String userName,
            String poolId) {
        try {
            AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
                    .username(userName)
                    .userPoolId(poolId)
                    .build();

            AdminGetUserResponse response =
 identityProviderClient.adminGetUser(userRequest);
            System.out.println("User status " + response.userStatusAsString());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken

- ConfirmDevice
- ConfirmSignUp
- InitiateAuth
- ListUsers
- ResendConfirmationCode
- RespondToAuthChallenge
- SignUp
- VerifySoftwareToken

使用 SDK for Java 2.x 的 Amazon Comprehend 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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从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

`CreateDocumentClassifier`

以下代码示例演示了如何使用 `CreateDocumentClassifier`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateDocumentClassifier中的。

`DetectDominantLanguage`

以下代码示例演示了如何使用 `DetectDominantLanguage`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DetectDominantLanguage中的。

DetectEntities

以下代码示例演示了如何使用 DetectEntities。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DetectEntities中的。

`DetectKeyPhrases`

以下代码示例演示了如何使用 `DetectKeyPhrases`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DetectKeyPhrases中的。

`DetectSentiment`

以下代码示例演示了如何使用 `DetectSentiment`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DetectSentiment中的。

`DetectSyntax`

以下代码示例演示了如何使用 `DetectSyntax`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DetectSyntax中的。

场景

构建 Amazon Lex 聊天机器人

以下代码示例演示如何创建用于吸引网站访客的聊天机器人。

适用于 Java 的 SDK 2.x

展示如何使用 Amazon Lex API 在 Web 应用程序中创建聊天机器人，以吸引网站访客。

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

创建消息收发应用程序

以下代码示例演示如何使用 Amazon SQS 创建消息收发应用程序。

适用于 Java 的 SDK 2.x

演示如何使用 Amazon SQS API 开发用于发送和检索消息的 Spring REST 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

- Amazon Comprehend
- Amazon SQS

创建用于分析客户反馈的应用程序

以下代码示例说明如何创建应用程序来分析客户意见卡、翻译其母语、确定其情绪并根据译后的文本生成音频文件。

适用于 Java 的 SDK 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

AWS Control Tower 使用适用于 Java 的 SDK 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 Control Tower。 AWS SDK for Java 2.x

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 基本功能
- 操作

开始使用

你好 AWS Control Tower

以下代码示例展示了如何开始使用 AWS Control Tower。

适用于 Java 的 SDK 2.x

> **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。

```
public class HelloControlTower {

    public static void main(String[] args) {
        try {
            ControlTowerClient controlTowerClient = ControlTowerClient.builder()
                .build() ;
            helloControlTower(controlTowerClient);
        } catch (ControlTowerException e) {
            System.out.println("Control Tower error occurred: " +
 e.awsErrorDetails().errorMessage());
        }
    }

    /**
     * Use the AWS SDK for Java (v2) to create an AWS Control Tower client
     * and list all available baselines.
     * This example uses the default settings specified in your shared credentials
     * and config files.
```

```
     *
     * @param controlTowerClient A ControlTowerClient object. This object wraps
     *                           the low-level AWS Control Tower service API.
     */
    public static void helloControlTower(ControlTowerClient controlTowerClient) {
        System.out.println("Hello, AWS Control Tower! Let's list available
 baselines:\n");

        ListBaselinesIterable paginator = controlTowerClient.listBaselinesPaginator(
                ListBaselinesRequest.builder().build());
        List<String> baselineNames = new ArrayList<>();

        try {
            paginator.stream()
                .flatMap(response -> response.baselines().stream())
                    .forEach(baseline -> baselineNames.add(baseline.name()));

            System.out.println(baselineNames.size() + " baseline(s) retrieved.");
            for (String baselineName : baselineNames) {
                System.out.println("\t" + baselineName);
            }

        } catch (ControlTowerException e) {
            if ("AccessDeniedException".equals(e.awsErrorDetails().errorCode())) {
                System.out.println("Access denied. Please ensure you have the
 necessary permissions.");
            } else {
                System.out.println("An error occurred: " + e.getMessage());
            }
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListBaselines中的。

基本功能

了解基本功能

以下代码示例展示了如何：

- 列出登录区。

- 列出、启用、获取、重置和禁用基准。
- 列出、启用、获取和禁用控件。

适用于 Java 的 SDK 2.x

> **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。

运行演示 AWS Control Tower 功能的交互式场景。

```
public class ControlTowerScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final Scanner scanner = new Scanner(in);

    private static OrganizationsClient orgClient;
    private static ControlCatalogClient catClient;

    private static String ouId = null;
    private static String ouArn = null;
    private static String landingZoneArn = null;
    private static boolean useLandingZone = false;

    private String stack = null;
    private String accountId = null;

    public static void main(String[] args) {

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS Control Tower basics scenario!");
        System.out.println(DASHES);

        try {
            runScenarioAsync();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

```
    // ----------------------------
    // Utilities
    // ----------------------------
    private static boolean askYesNo(String msg) {
        System.out.println(msg);
        return scanner.nextLine().trim().toLowerCase().startsWith("y");
    }

    private static void runScenarioAsync() {
        try {
            ControlTowerActions actions = new ControlTowerActions();

            // ----------------------------
            // Step 1: Landing Zones
            // ----------------------------
            System.out.println(DASHES);
            System.out.println("""
        Some demo operations require the use of a landing zone.
        You can use an existing landing zone or opt out of these operations in the
 demo.
        For instructions on how to set up a landing zone,
        see https://docs.aws.amazon.com/controltower/latest/userguide/getting-
started-from-console.html
        """);

            System.out.println("Step 1: Listing landing zones...");
            waitForInputToContinue(scanner);

            List<LandingZoneSummary> landingZones =
                    actions.listLandingZonesAsync().join();

            if (landingZones.isEmpty()) {
                System.out.println("No landing zones found. Landing-zone-dependent
 steps will be skipped.");
                useLandingZone = false;
                waitForInputToContinue(scanner);
            } else {
                System.out.println("\nAvailable Landing Zones:");
                for (int i = 0; i < landingZones.size(); i++) {
                    System.out.printf("%d) %s%n", i + 1, landingZones.get(i).arn());
                }

                if (askYesNo("Do you want to use the first landing zone in the list
 (" +
```

```
                        landingZones.get(0).arn() + ")? (y/n): ")) {
                    useLandingZone = true;
                    landingZoneArn = landingZones.get(0).arn();
                } else if (askYesNo("Do you want to use a different existing Landing
 Zone for this demo? (y/n): ")) {
                    useLandingZone = true;
                    System.out.println("Enter landing zone ARN: ");
                    landingZoneArn = scanner.nextLine().trim();
                } else {
                    System.out.println("Proceeding without a landing zone.");
                    useLandingZone = false;
                    waitForInputToContinue(scanner);
                }
            }

            // ----------------------------
            // Setup Organization + Sandbox OU
            // ----------------------------
            if (useLandingZone) {
                System.out.println("Using landing zone ARN: " + landingZoneArn);

                ControlTowerActions.OrgSetupResult result =
                        actions.setupOrganizationAsync().join();

                ouArn = result.sandboxOuArn();
                ouId  = result.sandboxOuArn();

                System.out.println("Organization ID: " + result.orgId());
                System.out.println("Using Sandbox OU ARN: " + ouArn);
            }

            // ----------------------------
            // Step 2: Baselines
            // ----------------------------
            System.out.println(DASHES);
            System.out.println("Step 2: Listing available baselines...");
            System.out.println("""
In this step, the program lists available AWS Control Tower baselines and may
 perform
baseline-related operations (enable, disable, reset) if requested.

NOTE:
AWS Control Tower enforces governance through baselines and mandatory controls
(guardrails). Mandatory controls are required for landing zone governance and may
```

```
restrict certain operations depending on the account, region, or organizational
 policy.

For more information, see:
- Types of baselines in AWS Control Tower:
  https://docs.aws.amazon.com/controltower/latest/userguide/types-of-baselines.html
- Mandatory controls (guardrails) in AWS Control Tower:
  https://docs.aws.amazon.com/controltower/latest/controlreference/mandatory-
controls.html
- Baseline API examples:
  https://docs.aws.amazon.com/controltower/latest/userguide/baseline-api-
examples.html
""");



            waitForInputToContinue(scanner);
            List<BaselineSummary> baselines =
                    actions.listBaselinesAsync().join();

            BaselineSummary controlTowerBaseline = null;
            for (BaselineSummary b : baselines) {
                System.out.println("Baseline: " + b.name());
                System.out.println("  ARN: " + b.arn());
                if ("AWSControlTowerBaseline".equals(b.name())) {
                    controlTowerBaseline = b;
                }
            }

            waitForInputToContinue(scanner);

            if (useLandingZone && controlTowerBaseline != null) {

                System.out.println("\nListing enabled baselines:");
                List<EnabledBaselineSummary> enabledBaselines =
                        actions.listEnabledBaselinesAsync().join();

                String enabledBaselineArn = null;
                for (EnabledBaselineSummary eb : enabledBaselines) {
                    System.out.println("Checking enabled baseline ARN: " +
 eb.arn());
                    if (eb.baselineIdentifier().equals(controlTowerBaseline.arn()))
 {
```

```
                        enabledBaselineArn = eb.arn(); // correct enabled ARN for
 this baseline
                        break; // stop after finding the matching one
                    }
                }

                if (enabledBaselineArn == null) {
                    System.out.println("No enabled baseline found for " +
 controlTowerBaseline.arn());
                } else {
                    System.out.println("Selected enabled baseline ARN for reset/
disable: " + enabledBaselineArn);
                }


                // Enable the Baseline
                if (askYesNo("Do you want to enable the Control Tower Baseline? (y/
n): ")) {
                    System.out.println("\nEnabling Control Tower Baseline...");

                    String baselineId = controlTowerBaseline.arn();
                    String enabledBaselineId =
                            actions.enableBaselineAsync(
                                    ouArn,                  // targetIdentifier #
 the OU or account ARN
                                    baselineId,             // baselineIdentifier #
 the Control Tower baseline ARN
                                    "5.0"                   // baselineVersion #
 version string
                            ).join();


                    System.out.println("Enabled baseline operation ID: " +
 enabledBaselineId);
                    if (enabledBaselineId == null) {
                       enabledBaselineId = enabledBaselineArn;
                    }

                    // Reset the Baseline
                    if (askYesNo("Do you want to reset the Control Tower Baseline?
 (y/n): ")) {
                        String operationId =

 actions.resetEnabledBaselineAsync(enabledBaselineId).join();
```

```
                        System.out.println("Reset baseline operation ID: " +
operationId);
                    }

                    if (askYesNo("Do you want to disable the Control Tower Baseline?
(y/n): ")) {
                        String operationId =

actions.disableBaselineAsync(enabledBaselineId).join();
                        System.out.println("Disabled baseline operation ID: " +
operationId);

                        System.out.println("Now we will re-enable the baseline and
wait 1 minute before making the call...");
                        try {
                            Thread.sleep(Duration.ofMinutes(1).toMillis());
                        } catch (InterruptedException e) {
                            Thread.currentThread().interrupt();
                            System.out.println("Wait interrupted");
                        }
                        String reEnabledBaselineId = actions.enableBaselineAsync(
                                ouArn,
                                baselineId,  // reuse baseline definition ARN
                                "5.0"
                        ).join();

                        System.out.println("Re-enabled baseline operation ID: " +
reEnabledBaselineId);
                    }
                }
            }

            // ----------------------------
            // Step 3: Controls
            // ----------------------------
            System.out.println(DASHES);
            System.out.println("Step 3: Managing Controls:");
            waitForInputToContinue(scanner);

            List<ControlSummary> controls =
                    actions.listControlsAsync().join();

            System.out.println("\nListing first 5 available Controls:");
            for (int i = 0; i < Math.min(5, controls.size()); i++) {
```

```
                ControlSummary c = controls.get(i);
                System.out.println("%d. %s - %s".formatted(i + 1, c.name(),
c.arn()));
            }

            if (useLandingZone) {
                waitForInputToContinue(scanner);

                List<EnabledControlSummary> enabledControls =
                        actions.listEnabledControlsAsync(ouArn).join();

                System.out.println("\nListing enabled controls:");
                for (int i = 0; i < enabledControls.size(); i++) {
                    System.out.println("%d. %s".formatted(i + 1,
enabledControls.get(i).controlIdentifier()));
                }

                String controlArnToEnable = null;
                for (ControlSummary control : controls) {
                    boolean enabled = enabledControls.stream()
                            .anyMatch(ec ->
ec.controlIdentifier().equals(control.arn()));
                    if (!enabled) {
                        controlArnToEnable = control.arn();
                        break;
                    }
                }

                waitForInputToContinue(scanner);
                if (controlArnToEnable != null &&
                        askYesNo("Do you want to enable the control " +
controlArnToEnable + "? (y/n): ")) {

                    String operationId =
                            actions.enableControlAsync(controlArnToEnable,
ouArn).join();

                    System.out.println("Enabled control with operation ID: " +
operationId);
                }

                waitForInputToContinue(scanner);

                if (controlArnToEnable != null &&
```

```
                        askYesNo("Do you want to disable the control? (y/n): ")) {

                    String operationId =
                            actions.disableControlAsync(controlArnToEnable,
 ouArn).join();

                    System.out.println("Disable operation ID: " + operationId);
                }
            }
            System.out.println("\nThis concludes the example scenario.");
            System.out.println("Thanks for watching!");
            System.out.println(DASHES);

        } catch (CompletionException e) {
            Throwable cause = e.getCause() != null ? e.getCause() : e;
            System.out.println("Scenario failed: " + cause.getMessage());
            throw e; // bubble up for tests / callers
        } catch (Exception e) {
            System.out.println("Unexpected error running scenario: " +
 e.getMessage());
            throw new RuntimeException(e);
        }
    }

    private static void waitForInputToContinue(Scanner sc) {
        System.out.println("\nEnter 'c' then <ENTER> to continue:");
        while (true) {
            String input = sc.nextLine();
            if ("c".equalsIgnoreCase(input.trim())) {
                System.out.println("Continuing...");
                break;
            }
        }
    }
}

public class ControlTowerActions {
    private static ControlCatalogAsyncClient controlCatalogAsyncClient;
    private static ControlTowerAsyncClient controlTowerAsyncClient;
    private static OrganizationsAsyncClient orgAsyncClient;

    private static OrganizationsAsyncClient getAsyncOrgClient() {
        if (orgAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
```

```
                    .maxConcurrency(50)
                    .connectionTimeout(Duration.ofSeconds(60))
                    .readTimeout(Duration.ofSeconds(60))
                    .writeTimeout(Duration.ofSeconds(60))
                    .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
                    .build();

            orgAsyncClient = OrganizationsAsyncClient.builder()
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
        return orgAsyncClient;
    }

    private static ControlCatalogAsyncClient getAsyncCatClient() {
        if (controlCatalogAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                    .maxConcurrency(100)
                    .connectionTimeout(Duration.ofSeconds(60))
                    .readTimeout(Duration.ofSeconds(60))
                    .writeTimeout(Duration.ofSeconds(60))
                    .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
                    .retryStrategy(RetryMode.STANDARD)
                    .build();

            controlCatalogAsyncClient = ControlCatalogAsyncClient.builder()
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
        return controlCatalogAsyncClient;
    }
```

```
    private static ControlTowerAsyncClient getAsyncClient() {
        if (controlTowerAsyncClient == null) {

            SdkAsyncHttpClient httpClient =
                    AwsCrtAsyncHttpClient.builder()
                            .maxConcurrency(100)
                            .connectionTimeout(Duration.ofSeconds(60))
                            .build();

            ClientOverrideConfiguration overrideConfig =
                    ClientOverrideConfiguration.builder()
                            .apiCallTimeout(Duration.ofMinutes(2))
                            .apiCallAttemptTimeout(Duration.ofSeconds(90))
                            .retryStrategy(RetryMode.STANDARD)
                            .build();

            controlTowerAsyncClient =
                    ControlTowerAsyncClient.builder()
                            .httpClient(httpClient)
                            .overrideConfiguration(overrideConfig)
                            .build();
        }

        return controlTowerAsyncClient;
    }

    public record OrgSetupResult(String orgId, String sandboxOuArn) {
    }

    public CompletableFuture<OrgSetupResult> setupOrganizationAsync() {
        System.out.println("Starting organization setup…");

        OrganizationsAsyncClient client = getAsyncOrgClient();

        // Step 1: Describe or create organization
        CompletableFuture<Organization> orgFuture = client.describeOrganization()
                .thenApply(desc -> {
                    System.out.println("Organization exists: " +
desc.organization().id());
                    return desc.organization();
                })
                .exceptionallyCompose(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;
                    if (cause instanceof AwsServiceException awsEx &&
```

```
"AWSOrganizationsNotInUseException".equals(awsEx.awsErrorDetails().errorCode())) {
                        System.out.println("No organization found. Creating one…");
                        return
client.createOrganization(CreateOrganizationRequest.builder()
                                        .featureSet(OrganizationFeatureSet.ALL)
                                        .build())
                                .thenApply(createResp -> {
                                    System.out.println("Created organization: {}" +
createResp.organization().id());
                                    return createResp.organization();
                                });
                    }
                    return CompletableFuture.failedFuture(
                            new CompletionException("Failed to describe or create
organization", cause)
                    );
                });

        // Step 2: Locate Sandbox OU
        return orgFuture.thenCompose(org -> {
            String orgId = org.id();
            System.out.println("Organization ID: {}" + orgId);

            return client.listRoots()
                    .thenCompose(rootsResp -> {
                        if (rootsResp.roots().isEmpty()) {
                            return CompletableFuture.failedFuture(
                                    new RuntimeException("No root found in
organization")
                            );
                        }
                        String rootId = rootsResp.roots().get(0).id();

                        ListOrganizationalUnitsForParentRequest ouRequest =
                                ListOrganizationalUnitsForParentRequest.builder()
                                        .parentId(rootId)
                                        .build();

                        ListOrganizationalUnitsForParentPublisher paginator =

client.listOrganizationalUnitsForParentPaginator(ouRequest);
```

```
                        AtomicReference<String> sandboxOuArnRef = new
AtomicReference<>();
                        return paginator.subscribe(page -> {
                                    for (OrganizationalUnit ou :
page.organizationalUnits()) {
                                        if ("Sandbox".equals(ou.name())) {
                                            sandboxOuArnRef.set(ou.arn());
                                            System.out.println("Found Sandbox OU: "
+ ou.id());
                                            break;
                                        }
                                    }
                                })
                                .thenApply(v -> {
                                    String sandboxArn = sandboxOuArnRef.get();
                                    if (sandboxArn == null) {
                                        System.out.println("Sandbox OU not found.");
                                    }
                                    return new OrgSetupResult(orgId, sandboxArn);
                                });
                    });
        }).exceptionally(ex -> {
            Throwable cause = ex.getCause() != null ? ex.getCause() : ex;
            System.out.println("Failed to setup organization: {}" +
cause.getMessage());
            throw new CompletionException(cause);
        });
    }

    /**
     * Lists all landing zones using pagination to retrieve complete results.
     *
     * @return a list of all landing zones
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<LandingZoneSummary>> listLandingZonesAsync() {
        System.out.println("Starting list landing zones paginator…");

        ListLandingZonesRequest request = ListLandingZonesRequest.builder().build();
        ListLandingZonesPublisher paginator =
getAsyncClient().listLandingZonesPaginator(request);
        List<LandingZoneSummary> landingZones = new ArrayList<>();
```

```
        return paginator.subscribe(response -> {
                    if (response.landingZones() != null && !
response.landingZones().isEmpty()) {
                        response.landingZones().forEach(lz -> {
                            System.out.println("Landing zone ARN: " + lz.arn());
                            landingZones.add(lz);
                        });
                    } else {
                        System.out.println("Page contained no landing zones.");
                    }
                })
                .thenRun(() -> System.out.println("Successfully retrieved "+
 landingZones.size() + " landing zones." ))
                .thenApply(v -> landingZones)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();
                        switch (errorCode) {
                            case "AccessDeniedException":
                                throw new CompletionException(
                                        "Access denied when listing landing zones: "
 + e.getMessage(), e);
                            default:
                                throw new CompletionException(
                                        "Error listing landing zones: " +
 e.getMessage(), e);
                        }
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing landing zones: " +
 cause.getMessage(), cause);
                    }

                    throw new CompletionException("Failed to list landing zones",
 cause);
                });
    }


    /**
```

```
     * Lists all available baselines using pagination to retrieve complete results.
     *
     * @return a list of all baselines
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<BaselineSummary>> listBaselinesAsync() {
        System.out.println("Starting list baselines paginator…");
        ListBaselinesRequest request = ListBaselinesRequest.builder().build();
        ListBaselinesPublisher paginator =
                getAsyncClient().listBaselinesPaginator(request);

        List<BaselineSummary> baselines = new ArrayList<>();
        return paginator.subscribe(response -> {
                    if (response.baselines() != null && !
response.baselines().isEmpty()) {
                        response.baselines().forEach(baseline -> {
                            baselines.add(baseline);
                        });
                    } else {
                        System.out.println("Page contained no baselines.");
                    }
                })
                .thenRun(() ->
                        System.out.println("Successfully listed baselines. Total: "
 + baselines.size())
                )
                .thenApply(v -> baselines)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("AccessDeniedException".equals(errorCode)) {
                            throw new CompletionException(
                                    "Access denied when listing baselines:
 %s".formatted(e.getMessage()),
                                    e
                            );
                        }

                        throw new CompletionException(
```

```
                                "Error listing baselines:
%s".formatted(e.getMessage()),
                                e
                        );
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing baselines:
%s".formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException("Failed to list baselines",
cause);
                });
    }

    /**
     * Lists all enabled baselines using pagination to retrieve complete results.
     *
     * @return a list of all enabled baselines
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<EnabledBaselineSummary>>
listEnabledBaselinesAsync() {
        System.out.println("Starting list enabled baselines paginator…");

        ListEnabledBaselinesRequest request =
                ListEnabledBaselinesRequest.builder().build();

        ListEnabledBaselinesPublisher paginator =
                getAsyncClient().listEnabledBaselinesPaginator(request);

        List<EnabledBaselineSummary> enabledBaselines = new ArrayList<>();
        return paginator.subscribe(response -> {
                    if (response.enabledBaselines() != null
                            && !response.enabledBaselines().isEmpty()) {

                        response.enabledBaselines().forEach(baseline -> {
                            enabledBaselines.add(baseline);
                        });
```

```
                    } else {
                        System.out.println("Page contained no enabled baselines.");
                    }
                })
                .thenRun(() ->
                        System.out.println(
                                "Successfully listed enabled baselines. Total: " +
                                        enabledBaselines.size()
                        )
                )
                .thenApply(v -> enabledBaselines)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("AccessDeniedException".equals(errorCode)) {
                            throw new CompletionException(
                                    "Access denied when listing enabled baselines:
%s".formatted(e.getMessage()), e);
                        }

                        throw new CompletionException(
                                "Error listing enabled baselines: %s"
                                        .formatted(e.getMessage()),
                                e
                        );
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing enabled baselines: %s"
                                        .formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException(
                            "Failed to list enabled baselines",
                            cause
                    );
                });
    }
```

```
   /**
    * Asynchronously enables a baseline for the specified target if not already
enabled.
    *
    * @param targetIdentifier       The ARN of the target (OU or account).
    * @param baselineIdentifier     The baseline definition ARN to enable.
    * @param baselineVersion        The baseline version to enable.
    * @return A CompletableFuture containing the enabled baseline ARN, or null if
already enabled.
    */
   public CompletableFuture<String> enableBaselineAsync(
           String targetIdentifier,
           String baselineIdentifier,
           String baselineVersion
   ) {
       EnableBaselineRequest request = EnableBaselineRequest.builder()
               .baselineIdentifier(baselineIdentifier)
               .baselineVersion(baselineVersion)
               .targetIdentifier(targetIdentifier)
               .build();

       return getAsyncClient().enableBaseline(request)
               .handle((resp, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                       if (cause instanceof ControlTowerException e) {
                           String code = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                           String msg = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();

                           if ("ValidationException".equals(code) &&
msg.contains("already enabled")) {
                               System.out.println("Baseline is already enabled for
this target # fetching ARN...");
                               return fetchEnabledBaselineArn(targetIdentifier,
baselineIdentifier)
                                       .join(); // fetch existing ARN synchronously
                           }
```

```
                            throw new RuntimeException("Error enabling baseline: " +
code + " - " + msg, e);
                        }

                        throw new RuntimeException("Unexpected error enabling
baseline: " + cause.getMessage(), cause);
                    }

                    return resp;
                })
                .thenCompose(result -> {
                    if (result instanceof EnableBaselineResponse resp) {
                        String operationId = resp.operationIdentifier();
                        String enabledBaselineArn = resp.arn();
                        System.out.println("Baseline enable started. ARN: " +
enabledBaselineArn
                                + ", operation ID: " + operationId);

                        // Inline polling
                        return CompletableFuture.supplyAsync(() -> {
                            while (true) {
                                GetBaselineOperationRequest opReq =
GetBaselineOperationRequest.builder()
                                        .operationIdentifier(operationId)
                                        .build();

                                GetBaselineOperationResponse opResp =
getAsyncClient().getBaselineOperation(opReq).join();
                                BaselineOperation op = opResp.baselineOperation();
                                BaselineOperationStatus status = op.status();
                                System.out.println("Operation " + operationId + "
status: " + status);

                                if (status == BaselineOperationStatus.SUCCEEDED) {
                                    return enabledBaselineArn;
                                } else if (status == BaselineOperationStatus.FAILED)
{
                                    String opId = op.operationIdentifier();
                                    String reason = op.statusMessage() != null ?
op.statusMessage() : "No failure reason provided";
                                    throw new RuntimeException("Baseline operation
failed (ID: " + opId + "), status: "
                                            + status + ", reason: " + reason);
                                }
```

```
                                try {
                                    Thread.sleep(Duration.ofSeconds(15).toMillis());
                                } catch (InterruptedException e) {
                                    Thread.currentThread().interrupt();
                                    throw new RuntimeException(e);
                                }
                            }
                        });
                    } else if (result instanceof String existingArn) {
                        // Already enabled branch
                        return CompletableFuture.completedFuture(existingArn);
                    }

                    return CompletableFuture.completedFuture(null);
                });
    }


    /**
     * Fetches the ARN of an already-enabled baseline for the target asynchronously.
     */
    private CompletableFuture<String> fetchEnabledBaselineArn(String
targetIdentifier, String baselineIdentifier) {
        return
getAsyncClient().listEnabledBaselines(ListEnabledBaselinesRequest.builder().build())
                .thenApply(listResp -> {
                    for (EnabledBaselineSummary eb : listResp.enabledBaselines()) {
                        if (baselineIdentifier.equals(eb.baselineIdentifier())
                                && targetIdentifier.equals(eb.targetIdentifier())) {
                            return eb.arn();
                        }
                    }
                    return null; // not yet available
                });
    }


    /**
     * Disables a baseline for a specified target.
     *
     * @param enabledBaselineIdentifier the identifier of the enabled baseline to
disable
     * @return the operation identifier
```

```
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> disableBaselineAsync(String
enabledBaselineIdentifier) {

        System.out.println("Starting disable of enabled baseline…");
        System.out.println("This operation will check the status every 15 seconds
until it completes (SUCCEEDED or FAILED).");

        DisableBaselineRequest request = DisableBaselineRequest.builder()
                .enabledBaselineIdentifier(enabledBaselineIdentifier)
                .build();

        return getAsyncClient().disableBaseline(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Disable baseline operation ID: " +
operationId);

                    // CompletableFuture that will be completed when operation
finishes
                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    // Polling loop
                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getBaselineOperationAsync(operationId)
                                    .thenAccept(statusObj -> {
                                        String status = statusObj.toString(); //
Convert enum/status to string for printing
                                        System.out.println("Current disable
operation status: " + status + " # waiting for SUCCEEDED or FAILED...");

                                        if ("SUCCEEDED".equalsIgnoreCase(status) ||
"FAILED".equalsIgnoreCase(status)) {
                                            System.out.println("Disable operation
finished with status: " + status);
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Schedule next poll in 15 seconds
```

```
                                            CompletableFuture.delayedExecutor(15,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        System.out.println("Error checking baseline
operation status: " + ex.getMessage());
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start first poll immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                        String errorMessage = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();

                        System.out.println("ControlTowerException caught while
disabling baseline: Code=" + errorCode + ", Message=" + errorMessage);
                        return null;
                    }

                    if (cause instanceof SdkException sdkEx) {
                        System.out.println("SDK exception caught while disabling
baseline: " + sdkEx.getMessage());
                        return null;
                    }

                    System.out.println("Unexpected exception while disabling
baseline: " + cause.getMessage());
                    return null;
                });
    }
```

```
/**
 * Gets the status of a baseline operation.
 *
 * @param operationIdentifier the identifier of the operation
 * @return the operation status
 * @throws ControlTowerException if a service-specific error occurs
 * @throws SdkException          if an SDK error occurs
 */
public CompletableFuture<BaselineOperationStatus> getBaselineOperationAsync(
        String operationIdentifier) {

    GetBaselineOperationRequest request = GetBaselineOperationRequest.builder()
            .operationIdentifier(operationIdentifier)
            .build();

    return getAsyncClient().getBaselineOperation(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause() != null
                            ? exception.getCause()
                            : exception;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("ResourceNotFoundException".equals(errorCode)) {
                            throw new CompletionException(
                                    "Baseline operation not found: %s"
                                            .formatted(e.getMessage()),
                                    e
                            );
                        }

                        throw new CompletionException(
                                "Error getting baseline operation status: %s"
                                        .formatted(e.getMessage()),
                                e
                        );
                    }

                    if (cause instanceof SdkException) {
```

```
                            throw new CompletionException(
                                    "SDK error getting baseline operation status:
%s"
                                            .formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to get baseline operation status",
                                cause
                        );
                    }
                })
                .thenApply(response -> {
                    BaselineOperationStatus status =
                            response.baselineOperation().status();
                    return status;
                });
    }


    /**
     * Lists all enabled controls for a specific target using pagination.
     *
     * @param targetIdentifier the identifier of the target (e.g., OU ARN)
     * @return a list of enabled controls
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<EnabledControlSummary>>
listEnabledControlsAsync(String targetIdentifier) {
        System.out.println("Starting list enabled controls paginator for target " +
targetIdentifier);
        ListEnabledControlsRequest request = ListEnabledControlsRequest.builder()
                .targetIdentifier(targetIdentifier)
                .build();

        ListEnabledControlsPublisher paginator =
getAsyncClient().listEnabledControlsPaginator(request);
        List<EnabledControlSummary> enabledControls = new ArrayList<>();

        // Subscribe to the paginator asynchronously
        return paginator.subscribe(response -> {
```

```
                    if (response.enabledControls() != null && !
response.enabledControls().isEmpty()) {
                        response.enabledControls().forEach(control -> {
                            enabledControls.add(control);
                        });
                    } else {
                        System.out.println("Page contained no enabled controls.");
                    }
                })
                .thenRun(() -> System.out.println(
                        "Successfully retrieved "+enabledControls.size() +" enabled
 controls for target "+targetIdentifier
                ))
                .thenApply(v -> enabledControls)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        switch (errorCode) {
                            case "AccessDeniedException":
                                throw new CompletionException(
                                        "Access denied when listing enabled
 controls: %s".formatted(e.getMessage()), e);

                            case "ResourceNotFoundException":
                                if (e.getMessage() != null &&
 e.getMessage().contains("not registered with AWS Control Tower")) {
                                    throw new CompletionException(
                                            "Control Tower must be enabled to work
 with controls", e);
                                }
                                throw new CompletionException(
                                        "Target not found when listing enabled
 controls: %s".formatted(e.getMessage()), e);

                            default:
                                throw new CompletionException(
                                        "Error listing enabled controls:
 %s".formatted(e.getMessage()), e);
                        }
                    }
```

```
                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing enabled controls:
%s".formatted(cause.getMessage()), cause);
                    }

                    throw new CompletionException("Failed to list enabled controls",
cause);
                });
    }


    /**
     * Enables a control for a specified target.
     *
     * @param controlIdentifier the identifier of the control to enable
     * @param targetIdentifier  the identifier of the target (e.g., OU ARN)
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> enableControlAsync(
            String controlIdentifier,
            String targetIdentifier) {

        EnableControlRequest request = EnableControlRequest.builder()
                .controlIdentifier(controlIdentifier)
                .targetIdentifier(targetIdentifier)
                .build();

        return getAsyncClient().enableControl(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Enable control operation started. Operation
ID: " + operationId);

                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getControlOperationAsync(operationId)
                                    .thenAccept(status -> {
```

```
                                        System.out.println("Control operation
status: " + status);

                                        if (status ==
ControlOperationStatus.SUCCEEDED
                                                || status ==
ControlOperationStatus.FAILED) {
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Poll again after 30 seconds
                                            CompletableFuture.delayedExecutor(30,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start polling immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();
                        String message = e.getMessage() != null ? e.getMessage() :
"";

                        if ("ValidationException".equals(errorCode)
                                && message.contains("already enabled")) {
                            System.out.println("Control is already enabled for this
target");
                            return null;
                        }

                        if ("ResourceNotFoundException".equals(errorCode)
```

```
                                && message.contains("not registered with AWS Control
Tower")) {
                            System.out.println(
                                    "Control Tower must be enabled to work with
controls.");
                            return null;
                        }

                        throw new CompletionException(
                                "Couldn't enable control: %s".formatted(message),
                                e
                        );
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error enabling control: %s"
                                        .formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException(
                            "Failed to enable control",
                            cause
                    );
                });
    }


    /**
     * Disables a control for a specified target.
     *
     * @param controlIdentifier the identifier of the control to disable
     * @param targetIdentifier  the identifier of the target (e.g., OU ARN)
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> disableControlAsync(
            String controlIdentifier,
            String targetIdentifier) {

        DisableControlRequest request = DisableControlRequest.builder()
```

```
                .controlIdentifier(controlIdentifier)
                .targetIdentifier(targetIdentifier)
                .build();

        return getAsyncClient().disableControl(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Disable control operation started. Operation
ID: " + operationId);

                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getControlOperationAsync(operationId)
                                    .thenAccept(status -> {
                                        System.out.println("Control operation
status: " + status);

                                        if (status ==
ControlOperationStatus.SUCCEEDED
                                                || status ==
ControlOperationStatus.FAILED) {
                                            resultFuture.complete(operationId);
                                        } else {
                                            // poll again after 30 seconds
                                            CompletableFuture.delayedExecutor(30,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // start polling immediately
                    poller.run();

                    return resultFuture;
```

```
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("ResourceNotFoundException".equals(errorCode)) {
                            // SPEC: notify user and continue
                            System.out.println("Control not found for disabling: " +
e.getMessage());
                            return null;
                        }

                        throw new CompletionException(
                                "Error disabling control: " + e.getMessage(), e);
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error disabling control: " +
cause.getMessage(), cause);
                    }

                    throw new CompletionException(
                            "Failed to disable control", cause);
                });
    }



    /**
     * Gets the status of a control operation.
     *
     * @param operationIdentifier the identifier of the operation
     * @return the operation status
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<ControlOperationStatus> getControlOperationAsync(
            String operationIdentifier) {

        GetControlOperationRequest request = GetControlOperationRequest.builder()
                .operationIdentifier(operationIdentifier)
```

```
                .build();

        return getAsyncClient().getControlOperation(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ControlTowerException e) {
                            String errorCode = e.awsErrorDetails().errorCode();

                            if ("ResourceNotFoundException".equals(errorCode)) {
                                throw new CompletionException(
                                        "Control operation not found:
%s".formatted(e.getMessage()),
                                        e
                                );
                            }

                            throw new CompletionException(
                                    "Error getting control operation status:
%s".formatted(e.getMessage()),
                                    e
                            );
                        }

                        if (cause instanceof SdkException) {
                            throw new CompletionException(
                                    "SDK error getting control operation status:
%s".formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException("Failed to get control
operation status", cause);
                    }
                })
                .thenApply(response -> response.controlOperation().status());
    }


    /**
     * Lists all controls in the Control Tower control catalog.
```

```
     *
     * @return a list of controls
     * @throws SdkException if a service-specific error occurs
     */
    public CompletableFuture<List<ControlSummary>> listControlsAsync() {
        System.out.println("Starting list controls paginator…");

        ListControlsRequest request = ListControlsRequest.builder().build();
        ListControlsPublisher paginator =
 getAsyncCatClient().listControlsPaginator(request);
        List<ControlSummary> controls = new ArrayList<>();

        return paginator.subscribe(response -> {
                    if (response.controls() != null && !
response.controls().isEmpty()) {
                        response.controls().forEach(control -> {
                            controls.add(control);
                        });
                    } else {
                        System.out.println("Page contained no controls.");
                    }
                })
                .thenRun(() -> System.out.println("Successfully retrieved " +
 controls.size() +" controls."))
                .thenApply(v -> controls)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof SdkException sdkEx) {
                        if (sdkEx.getMessage() != null &&
 sdkEx.getMessage().contains("AccessDeniedException")) {
                            throw new CompletionException(
                                    "Access denied when listing controls. Please
 ensure you have the necessary permissions.",
                                    sdkEx
                            );
                        } else {
                            throw new CompletionException(
                                    "SDK error listing controls:
 %s".formatted(sdkEx.getMessage()),
                                    sdkEx
                            );
                        }
                    }
```

```
                    throw new CompletionException("Failed to list controls", cause);
                });
    }

    /**
     * Resets an enabled baseline for a specific target.
     *
     * @param enabledBaselineIdentifier the identifier of the enabled baseline to
reset
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> resetEnabledBaselineAsync(String
enabledBaselineIdentifier) {

        System.out.println("Starting reset of enabled baseline…");
        System.out.println("This operation will check the status every 15 seconds
until it completes (SUCCEEDED or FAILED).");

        ResetEnabledBaselineRequest request = ResetEnabledBaselineRequest.builder()
                .enabledBaselineIdentifier(enabledBaselineIdentifier)
                .build();

        return getAsyncClient().resetEnabledBaseline(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Reset enabled baseline operation ID: " +
operationId);

                    // Polling loop
                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getBaselineOperationAsync(operationId)
                                    .thenAccept(statusObj -> {
                                        String status = statusObj.toString(); //
Convert enum/status to string for printing
                                        System.out.println("Current baseline
operation status: " + status + " # waiting for SUCCEEDED or FAILED...");
```

```
                                    if ("SUCCEEDED".equalsIgnoreCase(status) ||
"FAILED".equalsIgnoreCase(status)) {
                                        System.out.println("Baseline operation
finished with status: " + status);
                                        resultFuture.complete(operationId);
                                    } else {
                                        // Schedule next poll in 15 seconds
                                        CompletableFuture.delayedExecutor(15,
TimeUnit.SECONDS)
                                                .execute(this);
                                    }
                                })
                                .exceptionally(ex -> {
                                    System.out.println("Error checking baseline
operation status: " + ex.getMessage());
                                    resultFuture.completeExceptionally(ex);
                                    return null;
                                });
                        }
                    };

                    // Start first poll immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                        String errorMessage = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();

                        System.out.println("ControlTowerException caught: Code=" +
errorCode + ", Message=" + errorMessage);
                        return null;
                    }

                    if (cause instanceof SdkException sdkEx) {
                        System.out.println("SDK exception caught: " +
sdkEx.getMessage());
```

```
                        return null;
                    }

                    System.out.println("Unexpected exception resetting baseline: " +
 cause.getMessage());
                    return null;
                });
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - CreateLandingZone
  - DeleteLandingZone
  - DisableBaseline
  - DisableControl
  - EnableBaseline
  - EnableControl
  - GetControlOperation
  - GetLandingZoneOperation
  - ListBaselines
  - ListEnabledBaselines
  - ListEnabledControls
  - ListLandingZones
  - ResetEnabledBaseline

操作

`DisableBaseline`

以下代码示例演示了如何使用 `DisableBaseline`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Disables a baseline for a specified target.
    *
    * @param enabledBaselineIdentifier the identifier of the enabled baseline to
disable
    * @return the operation identifier
    * @throws ControlTowerException if a service-specific error occurs
    * @throws SdkException          if an SDK error occurs
    */
   public CompletableFuture<String> disableBaselineAsync(String
enabledBaselineIdentifier) {

       System.out.println("Starting disable of enabled baseline…");
       System.out.println("This operation will check the status every 15 seconds
until it completes (SUCCEEDED or FAILED).");

       DisableBaselineRequest request = DisableBaselineRequest.builder()
               .enabledBaselineIdentifier(enabledBaselineIdentifier)
               .build();

       return getAsyncClient().disableBaseline(request)
               .thenCompose(response -> {
                   String operationId = response.operationIdentifier();
                   System.out.println("Disable baseline operation ID: " +
operationId);

                   // CompletableFuture that will be completed when operation
finishes
                   CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                   // Polling loop
                   Runnable poller = new Runnable() {
```

```
                        @Override
                        public void run() {
                            getBaselineOperationAsync(operationId)
                                    .thenAccept(statusObj -> {
                                        String status = statusObj.toString(); //
Convert enum/status to string for printing
                                        System.out.println("Current disable
operation status: " + status + " # waiting for SUCCEEDED or FAILED...");

                                        if ("SUCCEEDED".equalsIgnoreCase(status) ||
"FAILED".equalsIgnoreCase(status)) {
                                            System.out.println("Disable operation
finished with status: " + status);
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Schedule next poll in 15 seconds
                                            CompletableFuture.delayedExecutor(15,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        System.out.println("Error checking baseline
operation status: " + ex.getMessage());
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start first poll immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                        String errorMessage = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();
```

```
                        System.out.println("ControlTowerException caught while
disabling baseline: Code=" + errorCode + ", Message=" + errorMessage);
                        return null;
                    }

                    if (cause instanceof SdkException sdkEx) {
                        System.out.println("SDK exception caught while disabling
baseline: " + sdkEx.getMessage());
                        return null;
                    }

                    System.out.println("Unexpected exception while disabling
baseline: " + cause.getMessage());
                    return null;
                });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DisableBaseline中的。

`DisableControl`

以下代码示例演示了如何使用 `DisableControl`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Disables a control for a specified target.
     *
     * @param controlIdentifier the identifier of the control to disable
     * @param targetIdentifier  the identifier of the target (e.g., OU ARN)
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
```

```
     */
    public CompletableFuture<String> disableControlAsync(
            String controlIdentifier,
            String targetIdentifier) {

        DisableControlRequest request = DisableControlRequest.builder()
                .controlIdentifier(controlIdentifier)
                .targetIdentifier(targetIdentifier)
                .build();

        return getAsyncClient().disableControl(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Disable control operation started. Operation
ID: " + operationId);

                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getControlOperationAsync(operationId)
                                    .thenAccept(status -> {
                                        System.out.println("Control operation
status: " + status);

                                        if (status ==
ControlOperationStatus.SUCCEEDED
                                                || status ==
ControlOperationStatus.FAILED) {
                                            resultFuture.complete(operationId);
                                        } else {
                                            // poll again after 30 seconds
                                            CompletableFuture.delayedExecutor(30,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
```

```
                    };

                    // start polling immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("ResourceNotFoundException".equals(errorCode)) {
                            // SPEC: notify user and continue
                            System.out.println("Control not found for disabling: " +
e.getMessage());
                            return null;
                        }

                        throw new CompletionException(
                                "Error disabling control: " + e.getMessage(), e);
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error disabling control: " +
cause.getMessage(), cause);
                    }

                    throw new CompletionException(
                            "Failed to disable control", cause);
                });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DisableControl中的。

EnableBaseline

以下代码示例演示了如何使用 EnableBaseline。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Asynchronously enables a baseline for the specified target if not already
enabled.
    *
    * @param targetIdentifier       The ARN of the target (OU or account).
    * @param baselineIdentifier     The baseline definition ARN to enable.
    * @param baselineVersion        The baseline version to enable.
    * @return A CompletableFuture containing the enabled baseline ARN, or null if
already enabled.
    */
   public CompletableFuture<String> enableBaselineAsync(
           String targetIdentifier,
           String baselineIdentifier,
           String baselineVersion
   ) {
       EnableBaselineRequest request = EnableBaselineRequest.builder()
               .baselineIdentifier(baselineIdentifier)
               .baselineVersion(baselineVersion)
               .targetIdentifier(targetIdentifier)
               .build();

       return getAsyncClient().enableBaseline(request)
               .handle((resp, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                       if (cause instanceof ControlTowerException e) {
                           String code = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                           String msg = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();
```

```
                            if ("ValidationException".equals(code) &&
msg.contains("already enabled")) {
                                System.out.println("Baseline is already enabled for
this target # fetching ARN...");
                                return fetchEnabledBaselineArn(targetIdentifier,
baselineIdentifier)
                                        .join(); // fetch existing ARN synchronously
                            }

                            throw new RuntimeException("Error enabling baseline: " +
code + " - " + msg, e);
                        }

                        throw new RuntimeException("Unexpected error enabling
baseline: " + cause.getMessage(), cause);
                    }

                    return resp;
                })
                .thenCompose(result -> {
                    if (result instanceof EnableBaselineResponse resp) {
                        String operationId = resp.operationIdentifier();
                        String enabledBaselineArn = resp.arn();
                        System.out.println("Baseline enable started. ARN: " +
enabledBaselineArn
                                + ", operation ID: " + operationId);

                        // Inline polling
                        return CompletableFuture.supplyAsync(() -> {
                            while (true) {
                                GetBaselineOperationRequest opReq =
GetBaselineOperationRequest.builder()
                                        .operationIdentifier(operationId)
                                        .build();

                                GetBaselineOperationResponse opResp =
getAsyncClient().getBaselineOperation(opReq).join();
                                BaselineOperation op = opResp.baselineOperation();
                                BaselineOperationStatus status = op.status();
                                System.out.println("Operation " + operationId + "
status: " + status);

                                if (status == BaselineOperationStatus.SUCCEEDED) {
                                    return enabledBaselineArn;
```

```
                                } else if (status == BaselineOperationStatus.FAILED)
{
                                    String opId = op.operationIdentifier();
                                    String reason = op.statusMessage() != null ?
op.statusMessage() : "No failure reason provided";
                                    throw new RuntimeException("Baseline operation
failed (ID: " + opId + "), status: "
                                            + status + ", reason: " + reason);
                                }

                                try {
                                    Thread.sleep(Duration.ofSeconds(15).toMillis());
                                } catch (InterruptedException e) {
                                    Thread.currentThread().interrupt();
                                    throw new RuntimeException(e);
                                }
                            }
                        });
                    } else if (result instanceof String existingArn) {
                        // Already enabled branch
                        return CompletableFuture.completedFuture(existingArn);
                    }

                    return CompletableFuture.completedFuture(null);
                });
    }


    /**
     * Fetches the ARN of an already-enabled baseline for the target asynchronously.
     */
    private CompletableFuture<String> fetchEnabledBaselineArn(String
targetIdentifier, String baselineIdentifier) {
        return
getAsyncClient().listEnabledBaselines(ListEnabledBaselinesRequest.builder().build())
                .thenApply(listResp -> {
                    for (EnabledBaselineSummary eb : listResp.enabledBaselines()) {
                        if (baselineIdentifier.equals(eb.baselineIdentifier())
                                && targetIdentifier.equals(eb.targetIdentifier())) {
                            return eb.arn();
                        }
                    }
                    return null; // not yet available
                });
```

```
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考EnableBaseline中的。

`EnableControl`

以下代码示例演示了如何使用 `EnableControl`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Enables a control for a specified target.
     *
     * @param controlIdentifier the identifier of the control to enable
     * @param targetIdentifier  the identifier of the target (e.g., OU ARN)
     * @return the operation identifier
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<String> enableControlAsync(
            String controlIdentifier,
            String targetIdentifier) {

        EnableControlRequest request = EnableControlRequest.builder()
                .controlIdentifier(controlIdentifier)
                .targetIdentifier(targetIdentifier)
                .build();

        return getAsyncClient().enableControl(request)
                .thenCompose(response -> {
                    String operationId = response.operationIdentifier();
                    System.out.println("Enable control operation started. Operation
ID: " + operationId);
```

```
                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getControlOperationAsync(operationId)
                                    .thenAccept(status -> {
                                        System.out.println("Control operation
status: " + status);

                                        if (status ==
ControlOperationStatus.SUCCEEDED
                                                || status ==
ControlOperationStatus.FAILED) {
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Poll again after 30 seconds
                                            CompletableFuture.delayedExecutor(30,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start polling immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();
                        String message = e.getMessage() != null ? e.getMessage() :
"";

                        if ("ValidationException".equals(errorCode)
```

```
                                && message.contains("already enabled")) {
                            System.out.println("Control is already enabled for this
target");
                            return null;
                        }

                        if ("ResourceNotFoundException".equals(errorCode)
                                && message.contains("not registered with AWS Control
Tower")) {
                            System.out.println(
                                    "Control Tower must be enabled to work with
controls.");
                            return null;
                        }

                        throw new CompletionException(
                                "Couldn't enable control: %s".formatted(message),
                                e
                        );
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error enabling control: %s"
                                        .formatted(cause.getMessage()),
                                cause
                        );
                    }

                    throw new CompletionException(
                            "Failed to enable control",
                            cause
                    );
                });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考EnableControl中的。

GetBaselineOperation

以下代码示例演示了如何使用 GetBaselineOperation。

适用于 Java 的 SDK 2.x

> **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。

```
/**
 * Gets the status of a baseline operation.
 *
 * @param operationIdentifier the identifier of the operation
 * @return the operation status
 * @throws ControlTowerException if a service-specific error occurs
 * @throws SdkException          if an SDK error occurs
 */
public CompletableFuture<BaselineOperationStatus> getBaselineOperationAsync(
        String operationIdentifier) {

    GetBaselineOperationRequest request = GetBaselineOperationRequest.builder()
            .operationIdentifier(operationIdentifier)
            .build();

    return getAsyncClient().getBaselineOperation(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause() != null
                            ? exception.getCause()
                            : exception;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("ResourceNotFoundException".equals(errorCode)) {
                            throw new CompletionException(
                                    "Baseline operation not found: %s"
                                            .formatted(e.getMessage()),
                                    e
                            );
                        }
```

```
                            throw new CompletionException(
                                    "Error getting baseline operation status: %s"
                                            .formatted(e.getMessage()),
                                    e
                            );
                        }

                        if (cause instanceof SdkException) {
                            throw new CompletionException(
                                    "SDK error getting baseline operation status:
%s"
                                            .formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to get baseline operation status",
                                cause
                        );
                    }
                })
                .thenApply(response -> {
                    BaselineOperationStatus status =
                            response.baselineOperation().status();
                    return status;
                });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[GetBaselineOperation](#)中的。

`GetControlOperation`

以下代码示例演示了如何使用 `GetControlOperation`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Gets the status of a control operation.
    *
    * @param operationIdentifier the identifier of the operation
    * @return the operation status
    * @throws ControlTowerException if a service-specific error occurs
    * @throws SdkException          if an SDK error occurs
    */
   public CompletableFuture<ControlOperationStatus> getControlOperationAsync(
           String operationIdentifier) {

       GetControlOperationRequest request = GetControlOperationRequest.builder()
               .operationIdentifier(operationIdentifier)
               .build();

       return getAsyncClient().getControlOperation(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                       if (cause instanceof ControlTowerException e) {
                           String errorCode = e.awsErrorDetails().errorCode();

                           if ("ResourceNotFoundException".equals(errorCode)) {
                               throw new CompletionException(
                                       "Control operation not found:
%s".formatted(e.getMessage()),
                                       e
                               );
                           }

                           throw new CompletionException(
                                   "Error getting control operation status:
%s".formatted(e.getMessage()),
                                   e
                           );
                       }

                       if (cause instanceof SdkException) {
                           throw new CompletionException(
```

```
                                    "SDK error getting control operation status:
%s".formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException("Failed to get control
operation status", cause);
                    }
                })
                .thenApply(response -> response.controlOperation().status());
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetControlOperation中的。

`ListBaselines`

以下代码示例演示了如何使用 `ListBaselines`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Lists all available baselines using pagination to retrieve complete results.
     *
     * @return a list of all baselines
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<BaselineSummary>> listBaselinesAsync() {
        System.out.println("Starting list baselines paginator…");
        ListBaselinesRequest request = ListBaselinesRequest.builder().build();
        ListBaselinesPublisher paginator =
                getAsyncClient().listBaselinesPaginator(request);
```

```
        List<BaselineSummary> baselines = new ArrayList<>();
        return paginator.subscribe(response -> {
                    if (response.baselines() != null && !
response.baselines().isEmpty()) {
                        response.baselines().forEach(baseline -> {
                            baselines.add(baseline);
                        });
                    } else {
                        System.out.println("Page contained no baselines.");
                    }
                })
                .thenRun(() ->
                        System.out.println("Successfully listed baselines. Total: "
 + baselines.size())
                )
                .thenApply(v -> baselines)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        if ("AccessDeniedException".equals(errorCode)) {
                            throw new CompletionException(
                                    "Access denied when listing baselines:
 %s".formatted(e.getMessage()),
                                    e
                            );
                        }

                        throw new CompletionException(
                                "Error listing baselines:
 %s".formatted(e.getMessage()),
                                e
                        );
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing baselines:
 %s".formatted(cause.getMessage()),
                                cause
                        );
                    }
```

```
                    throw new CompletionException("Failed to list baselines",
cause);
                });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListBaselines中的。

`ListEnabledBaselines`

以下代码示例演示了如何使用 `ListEnabledBaselines`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Lists all enabled baselines using pagination to retrieve complete results.
     *
     * @return a list of all enabled baselines
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<EnabledBaselineSummary>>
listEnabledBaselinesAsync() {
        System.out.println("Starting list enabled baselines paginator…");

        ListEnabledBaselinesRequest request =
                ListEnabledBaselinesRequest.builder().build();

        ListEnabledBaselinesPublisher paginator =
                getAsyncClient().listEnabledBaselinesPaginator(request);

        List<EnabledBaselineSummary> enabledBaselines = new ArrayList<>();
        return paginator.subscribe(response -> {
                    if (response.enabledBaselines() != null
                            && !response.enabledBaselines().isEmpty()) {
```

```
                    response.enabledBaselines().forEach(baseline -> {
                        enabledBaselines.add(baseline);
                    });
                } else {
                    System.out.println("Page contained no enabled baselines.");
                }
            })
            .thenRun(() ->
                    System.out.println(
                            "Successfully listed enabled baselines. Total: " +
                                    enabledBaselines.size()
                    )
            )
            .thenApply(v -> enabledBaselines)
            .exceptionally(ex -> {
                Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                if (cause instanceof ControlTowerException e) {
                    String errorCode = e.awsErrorDetails().errorCode();

                    if ("AccessDeniedException".equals(errorCode)) {
                        throw new CompletionException(
                                "Access denied when listing enabled baselines:
%s".formatted(e.getMessage()), e);
                    }

                    throw new CompletionException(
                            "Error listing enabled baselines: %s"
                                    .formatted(e.getMessage()),
                            e
                    );
                }

                if (cause instanceof SdkException) {
                    throw new CompletionException(
                            "SDK error listing enabled baselines: %s"
                                    .formatted(cause.getMessage()),
                            cause
                    );
                }

                throw new CompletionException(
                        "Failed to list enabled baselines",
```

```
                            cause
                    );
                });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListEnabledBaselines中的。

`ListEnabledControls`

以下代码示例演示了如何使用 `ListEnabledControls`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Lists all enabled controls for a specific target using pagination.
     *
     * @param targetIdentifier the identifier of the target (e.g., OU ARN)
     * @return a list of enabled controls
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<EnabledControlSummary>>
listEnabledControlsAsync(String targetIdentifier) {
        System.out.println("Starting list enabled controls paginator for target " +
targetIdentifier);
        ListEnabledControlsRequest request = ListEnabledControlsRequest.builder()
                .targetIdentifier(targetIdentifier)
                .build();

        ListEnabledControlsPublisher paginator =
getAsyncClient().listEnabledControlsPaginator(request);
        List<EnabledControlSummary> enabledControls = new ArrayList<>();

        // Subscribe to the paginator asynchronously
```

```
        return paginator.subscribe(response -> {
                    if (response.enabledControls() != null && !
response.enabledControls().isEmpty()) {
                        response.enabledControls().forEach(control -> {
                            enabledControls.add(control);
                        });
                    } else {
                        System.out.println("Page contained no enabled controls.");
                    }
                })
                .thenRun(() -> System.out.println(
                        "Successfully retrieved "+enabledControls.size() +" enabled
 controls for target "+targetIdentifier
                ))
                .thenApply(v -> enabledControls)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();

                        switch (errorCode) {
                            case "AccessDeniedException":
                                throw new CompletionException(
                                        "Access denied when listing enabled
 controls: %s".formatted(e.getMessage()), e);

                            case "ResourceNotFoundException":
                                if (e.getMessage() != null &&
 e.getMessage().contains("not registered with AWS Control Tower")) {
                                    throw new CompletionException(
                                            "Control Tower must be enabled to work
 with controls", e);
                                }
                                throw new CompletionException(
                                        "Target not found when listing enabled
 controls: %s".formatted(e.getMessage()), e);

                            default:
                                throw new CompletionException(
                                        "Error listing enabled controls:
 %s".formatted(e.getMessage()), e);
                        }
                    }
```

```
                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing enabled controls:
%s".formatted(cause.getMessage()), cause);
                    }

                    throw new CompletionException("Failed to list enabled controls",
cause);
                });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListEnabledControls中的。

`ListLandingZones`

以下代码示例演示了如何使用 `ListLandingZones`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Lists all landing zones using pagination to retrieve complete results.
     *
     * @return a list of all landing zones
     * @throws ControlTowerException if a service-specific error occurs
     * @throws SdkException          if an SDK error occurs
     */
    public CompletableFuture<List<LandingZoneSummary>> listLandingZonesAsync() {
        System.out.println("Starting list landing zones paginator…");

        ListLandingZonesRequest request = ListLandingZonesRequest.builder().build();
        ListLandingZonesPublisher paginator =
getAsyncClient().listLandingZonesPaginator(request);
        List<LandingZoneSummary> landingZones = new ArrayList<>();
```

```
        return paginator.subscribe(response -> {
                    if (response.landingZones() != null && !
response.landingZones().isEmpty()) {
                        response.landingZones().forEach(lz -> {
                            System.out.println("Landing zone ARN: " + lz.arn());
                            landingZones.add(lz);
                        });
                    } else {
                        System.out.println("Page contained no landing zones.");
                    }
                })
                .thenRun(() -> System.out.println("Successfully retrieved "+
 landingZones.size() + " landing zones." ))
                .thenApply(v -> landingZones)
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
                        String errorCode = e.awsErrorDetails().errorCode();
                        switch (errorCode) {
                            case "AccessDeniedException":
                                throw new CompletionException(
                                        "Access denied when listing landing zones: "
 + e.getMessage(), e);
                            default:
                                throw new CompletionException(
                                        "Error listing landing zones: " +
 e.getMessage(), e);
                        }
                    }

                    if (cause instanceof SdkException) {
                        throw new CompletionException(
                                "SDK error listing landing zones: " +
 cause.getMessage(), cause);
                    }

                    throw new CompletionException("Failed to list landing zones",
 cause);
                });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListLandingZones中的。

`ResetEnabledBaseline`

以下代码示例演示了如何使用 `ResetEnabledBaseline`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Resets an enabled baseline for a specific target.
    *
    * @param enabledBaselineIdentifier the identifier of the enabled baseline to
reset
    * @return the operation identifier
    * @throws ControlTowerException if a service-specific error occurs
    * @throws SdkException          if an SDK error occurs
    */
   public CompletableFuture<String> resetEnabledBaselineAsync(String
enabledBaselineIdentifier) {

       System.out.println("Starting reset of enabled baseline…");
       System.out.println("This operation will check the status every 15 seconds
until it completes (SUCCEEDED or FAILED).");

       ResetEnabledBaselineRequest request = ResetEnabledBaselineRequest.builder()
               .enabledBaselineIdentifier(enabledBaselineIdentifier)
               .build();

       return getAsyncClient().resetEnabledBaseline(request)
               .thenCompose(response -> {
                   String operationId = response.operationIdentifier();
                   System.out.println("Reset enabled baseline operation ID: " +
operationId);

                   // Polling loop
```

```
                    CompletableFuture<String> resultFuture = new
CompletableFuture<>();

                    Runnable poller = new Runnable() {
                        @Override
                        public void run() {
                            getBaselineOperationAsync(operationId)
                                    .thenAccept(statusObj -> {
                                        String status = statusObj.toString(); //
Convert enum/status to string for printing
                                        System.out.println("Current baseline
operation status: " + status + " # waiting for SUCCEEDED or FAILED...");

                                        if ("SUCCEEDED".equalsIgnoreCase(status) ||
"FAILED".equalsIgnoreCase(status)) {
                                            System.out.println("Baseline operation
finished with status: " + status);
                                            resultFuture.complete(operationId);
                                        } else {
                                            // Schedule next poll in 15 seconds
                                            CompletableFuture.delayedExecutor(15,
TimeUnit.SECONDS)
                                                    .execute(this);
                                        }
                                    })
                                    .exceptionally(ex -> {
                                        System.out.println("Error checking baseline
operation status: " + ex.getMessage());
                                        resultFuture.completeExceptionally(ex);
                                        return null;
                                    });
                        }
                    };

                    // Start first poll immediately
                    poller.run();

                    return resultFuture;
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ControlTowerException e) {
```

```
                        String errorCode = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorCode() : "UNKNOWN";
                        String errorMessage = e.awsErrorDetails() != null ?
e.awsErrorDetails().errorMessage() : e.getMessage();

                        System.out.println("ControlTowerException caught: Code=" +
errorCode + ", Message=" + errorMessage);
                        return null;
                    }

                    if (cause instanceof SdkException sdkEx) {
                        System.out.println("SDK exception caught: " +
sdkEx.getMessage());
                        return null;
                    }

                    System.out.println("Unexpected exception resetting baseline: " +
cause.getMessage());
                    return null;
                });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ResetEnabledBaseline中的。

使用适用于 Java 的 SDK 2.x 的 Firehose 示例

以下代码示例向您展示了如何使用 AWS SDK for Java 2.x 与 Firehose 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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从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

`PutRecord`

以下代码示例演示了如何使用 `PutRecord`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Puts a record to the specified Amazon Kinesis Data Firehose delivery stream.
    *
    * @param record The record to be put to the delivery stream. The record must be
a {@link Map} of String keys and Object values.
    * @param deliveryStreamName The name of the Amazon Kinesis Data Firehose
delivery stream to which the record should be put.
    * @throws IllegalArgumentException if the input record or delivery stream name
is null or empty.
    * @throws RuntimeException if there is an error putting the record to the
delivery stream.
    */
   public static void putRecord(Map<String, Object> record, String
deliveryStreamName) {
       if (record == null || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
           throw new IllegalArgumentException("Invalid input: record or delivery
stream name cannot be null/empty");
       }
       try {
           String jsonRecord = new ObjectMapper().writeValueAsString(record);
           Record firehoseRecord = Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
               .build();
```

```
            PutRecordRequest putRecordRequest = PutRecordRequest.builder()
                .deliveryStreamName(deliveryStreamName)
                .record(firehoseRecord)
                .build();

            getFirehoseClient().putRecord(putRecordRequest);
            System.out.println("Record sent: " + jsonRecord);
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record: " + e.getMessage(),
e);
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutRecord中的。

PutRecordBatch

以下代码示例演示了如何使用 PutRecordBatch。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Puts a batch of records to an Amazon Kinesis Data Firehose delivery stream.
     *
     * @param records           a list of maps representing the records to be sent
     * @param batchSize         the maximum number of records to include in each
batch
     * @param deliveryStreamName the name of the Kinesis Data Firehose delivery
stream
     * @throws IllegalArgumentException if the input parameters are invalid (null or
empty)
     * @throws RuntimeException         if there is an error putting the record
batch
     */
```

```
   public static void putRecordBatch(List<Map<String, Object>> records, int
batchSize, String deliveryStreamName) {
       if (records == null || records.isEmpty() || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
           throw new IllegalArgumentException("Invalid input: records or delivery
stream name cannot be null/empty");
       }
       ObjectMapper objectMapper = new ObjectMapper();

       try {
           for (int i = 0; i < records.size(); i += batchSize) {
               List<Map<String, Object>> batch = records.subList(i, Math.min(i +
batchSize, records.size()));

               List<Record> batchRecords = batch.stream().map(record -> {
                   try {
                       String jsonRecord = objectMapper.writeValueAsString(record);
                       return Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                           .build();
                   } catch (Exception e) {
                       throw new RuntimeException("Error creating Firehose record",
e);
                   }
               }).collect(Collectors.toList());

               PutRecordBatchRequest request = PutRecordBatchRequest.builder()
                   .deliveryStreamName(deliveryStreamName)
                   .records(batchRecords)
                   .build();

               PutRecordBatchResponse response =
getFirehoseClient().putRecordBatch(request);

               if (response.failedPutCount() > 0) {
                   response.requestResponses().stream()
                       .filter(r -> r.errorCode() != null)
                       .forEach(r -> System.err.println("Failed record: " +
r.errorMessage()));
               }
               System.out.println("Batch sent with size: " + batchRecords.size());
           }
       } catch (Exception e) {
```

```
            throw new RuntimeException("Failed to put record batch: " +
 e.getMessage(), e);
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutRecordBatch中的。

场景

将记录放入 Firehose

以下代码示例演示如何使用 Firehose 处理单个记录和批量记录。

适用于 Java 的 SDK 2.x

> **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。

此示例将单个记录和批量记录放入 Firehose。

```
/**
 * Amazon Firehose Scenario example using Java V2 SDK.
 *
 * Demonstrates individual and batch record processing,
 * and monitoring Firehose delivery stream metrics.
 */
public class FirehoseScenario {

    private static FirehoseClient firehoseClient;
    private static CloudWatchClient cloudWatchClient;

    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <deliveryStreamName>
                Where:
                    deliveryStreamName - The Firehose delivery stream name.
                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            return;
        }

        String deliveryStreamName = args[0];

        try {
            // Read and parse sample data.
            String jsonContent = readJsonFile("sample_records.json");
            ObjectMapper objectMapper = new ObjectMapper();
            List<Map<String, Object>> sampleData =
objectMapper.readValue(jsonContent, new TypeReference<>() {});

            // Process individual records.
            System.out.println("Processing individual records...");
            sampleData.subList(0, 100).forEach(record -> {
                try {
                    putRecord(record, deliveryStreamName);
                } catch (Exception e) {
                    System.err.println("Error processing record: " +
e.getMessage());
                }
            });

            // Monitor metrics.
            monitorMetrics(deliveryStreamName);

            // Process batch records.
            System.out.println("Processing batch records...");
            putRecordBatch(sampleData.subList(100, sampleData.size()), 500,
deliveryStreamName);
            monitorMetrics(deliveryStreamName);

        } catch (Exception e) {
            System.err.println("Scenario failed: " + e.getMessage());
        } finally {
            closeClients();
        }
    }

    private static FirehoseClient getFirehoseClient() {
        if (firehoseClient == null) {
```

```
            firehoseClient = FirehoseClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return firehoseClient;
    }

    private static CloudWatchClient getCloudWatchClient() {
        if (cloudWatchClient == null) {
            cloudWatchClient = CloudWatchClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return cloudWatchClient;
    }

    /**
     * Puts a record to the specified Amazon Kinesis Data Firehose delivery stream.
     *
     * @param record The record to be put to the delivery stream. The record must be
a {@link Map} of String keys and Object values.
     * @param deliveryStreamName The name of the Amazon Kinesis Data Firehose
delivery stream to which the record should be put.
     * @throws IllegalArgumentException if the input record or delivery stream name
is null or empty.
     * @throws RuntimeException if there is an error putting the record to the
delivery stream.
     */
    public static void putRecord(Map<String, Object> record, String
deliveryStreamName) {
        if (record == null || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
            throw new IllegalArgumentException("Invalid input: record or delivery
stream name cannot be null/empty");
        }
        try {
            String jsonRecord = new ObjectMapper().writeValueAsString(record);
            Record firehoseRecord = Record.builder()

.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                .build();

            PutRecordRequest putRecordRequest = PutRecordRequest.builder()
                .deliveryStreamName(deliveryStreamName)
```

```
                .record(firehoseRecord)
                .build();

            getFirehoseClient().putRecord(putRecordRequest);
            System.out.println("Record sent: " + jsonRecord);
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record: " + e.getMessage(),
e);
        }
    }


    /**
     * Puts a batch of records to an Amazon Kinesis Data Firehose delivery stream.
     *
     * @param records           a list of maps representing the records to be sent
     * @param batchSize         the maximum number of records to include in each
batch
     * @param deliveryStreamName the name of the Kinesis Data Firehose delivery
stream
     * @throws IllegalArgumentException if the input parameters are invalid (null or
empty)
     * @throws RuntimeException         if there is an error putting the record
batch
     */
    public static void putRecordBatch(List<Map<String, Object>> records, int
batchSize, String deliveryStreamName) {
        if (records == null || records.isEmpty() || deliveryStreamName == null ||
deliveryStreamName.isEmpty()) {
            throw new IllegalArgumentException("Invalid input: records or delivery
stream name cannot be null/empty");
        }
        ObjectMapper objectMapper = new ObjectMapper();

        try {
            for (int i = 0; i < records.size(); i += batchSize) {
                List<Map<String, Object>> batch = records.subList(i, Math.min(i +
batchSize, records.size()));

                List<Record> batchRecords = batch.stream().map(record -> {
                    try {
                        String jsonRecord = objectMapper.writeValueAsString(record);
                        return Record.builder()
```

```
.data(SdkBytes.fromByteArray(jsonRecord.getBytes(StandardCharsets.UTF_8)))
                            .build();
                    } catch (Exception e) {
                        throw new RuntimeException("Error creating Firehose record",
e);
                    }
                }).collect(Collectors.toList());

                PutRecordBatchRequest request = PutRecordBatchRequest.builder()
                    .deliveryStreamName(deliveryStreamName)
                    .records(batchRecords)
                    .build();

                PutRecordBatchResponse response =
getFirehoseClient().putRecordBatch(request);

                if (response.failedPutCount() > 0) {
                    response.requestResponses().stream()
                        .filter(r -> r.errorCode() != null)
                        .forEach(r -> System.err.println("Failed record: " +
r.errorMessage()));
                }
                System.out.println("Batch sent with size: " + batchRecords.size());
            }
        } catch (Exception e) {
            throw new RuntimeException("Failed to put record batch: " +
e.getMessage(), e);
        }
    }

    public static void monitorMetrics(String deliveryStreamName) {
        Instant endTime = Instant.now();
        Instant startTime = endTime.minusSeconds(600);

        List<String> metrics = List.of("IncomingBytes", "IncomingRecords",
"FailedPutCount");
        metrics.forEach(metric -> monitorMetric(metric, startTime, endTime,
deliveryStreamName));
    }

    private static void monitorMetric(String metricName, Instant startTime, Instant
endTime, String deliveryStreamName) {
        try {
```

```
            GetMetricStatisticsRequest request =
 GetMetricStatisticsRequest.builder()
                .namespace("AWS/Firehose")
                .metricName(metricName)

 .dimensions(Dimension.builder().name("DeliveryStreamName").value(deliveryStreamName).build(
                .startTime(startTime)
                .endTime(endTime)
                .period(60)
                .statistics(Statistic.SUM)
                .build();

            GetMetricStatisticsResponse response =
 getCloudWatchClient().getMetricStatistics(request);
            double totalSum =
 response.datapoints().stream().mapToDouble(Datapoint::sum).sum();
            System.out.println(metricName + ": " + totalSum);
        } catch (Exception e) {
            System.err.println("Failed to monitor metric " + metricName + ": " +
 e.getMessage());
        }
    }

    public static String readJsonFile(String fileName) throws IOException {
        try (InputStream inputStream =
 FirehoseScenario.class.getResourceAsStream("/" + fileName);
             Scanner scanner = new Scanner(inputStream, StandardCharsets.UTF_8)) {
            return scanner.useDelimiter("\\\\A").next();
        } catch (Exception e) {
            throw new RuntimeException("Error reading file: " + fileName, e);
        }
    }

    private static void closeClients() {
        try {
            if (firehoseClient != null) firehoseClient.close();
            if (cloudWatchClient != null) cloudWatchClient.close();
        } catch (Exception e) {
            System.err.println("Error closing clients: " + e.getMessage());
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - PutRecord
  - PutRecordBatch

使用适用于 Java 的 SDK 2.x 的 Amazon DocumentDB 示例

以下代码示例向您展示了如何在 Amazon DocumentDB 中 AWS SDK for Java 2.x 使用来执行操作和实现常见场景。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 无服务器示例

无服务器示例

通过 Amazon DocumentDB 触发器调用 Lambda 函数

以下代码示例演示如何实现一个 Lambda 函数，该函数接收通过接收来自 DocumentDB 更改流的记录而触发的事件。该函数检索 DocumentDB 有效负载，并记录下记录内容。

适用于 Java 的 SDK 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 Amazon DocumentDB 事件与 Lambda 结合使用。

```
import java.util.List;
import java.util.Map;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
```

```
public class Example implements RequestHandler<Map<String, Object>, String> {

    @SuppressWarnings("unchecked")
    @Override
    public String handleRequest(Map<String, Object> event, Context context) {
        List<Map<String, Object>> events = (List<Map<String, Object>>)
 event.get("events");
        for (Map<String, Object> record : events) {
            Map<String, Object> eventData = (Map<String, Object>)
 record.get("event");
            processEventData(eventData);
        }

        return "OK";
    }

    @SuppressWarnings("unchecked")
    private void processEventData(Map<String, Object> eventData) {
        String operationType = (String) eventData.get("operationType");
        System.out.println("operationType: %s".formatted(operationType));

        Map<String, Object> ns = (Map<String, Object>) eventData.get("ns");

        String db = (String) ns.get("db");
        System.out.println("db: %s".formatted(db));
        String coll = (String) ns.get("coll");
        System.out.println("coll: %s".formatted(coll));

        Map<String, Object> fullDocument = (Map<String, Object>)
 eventData.get("fullDocument");
        System.out.println("fullDocument: %s".formatted(fullDocument));
    }

}
```

使用 SDK for Java 2.x 的 DynamoDB 示例

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

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

场景是向您展示如何通过在一个服务中调用多个函数或与其他 AWS 服务服务结合来完成特定任务的代码示例。

AWS 社区贡献就是由多个团队创建和维护的示例 AWS。要提供反馈，请使用链接存储库中提供的机制。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 基本功能
- 操作
- 场景
- 无服务器示例
- AWS 社区捐款

开始使用

开始使用 DynamoDB

以下代码示例显示如何开始使用 DynamoDB。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListTables中的。

基本功能

了解基本功能

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。
- 向 JSON 示例文件的表中写入电影数据。
- 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。

适用于 Java 的 SDK 2.x

> **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。

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)
```

```
            .billingMode(BillingMode.PAY_PER_REQUEST) //  DynamoDB automatically
scales based on traffic.
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            String newTable = response.tableDescription().tableName();
            System.out.println("The " + newTable + " was successfully created.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

创建帮助函数以下载并提取示例 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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 * <p>
 * This Java example performs these tasks:
 * <p>
 * 1. Creates the Amazon DynamoDB Movie table with partition and sort key.
 * 2. Puts data into the Amazon DynamoDB table from a JSON document using the
 * Enhanced client.
 * 3. Gets data from the Movie table.
 * 4. Adds a new item.
 * 5. Updates an item.
 * 6. Uses a Scan to query items using the Enhanced client.
 * 7. Queries all items where the year is 2013 using the Enhanced Client.
 * 8. Deletes the table.
 */
```

```
public class Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws IOException {
        String tableName = "Movies";
        String fileName = "../../../resources/sample_files/movies.json";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon DynamoDB example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
            "1. Creating an Amazon DynamoDB table named Movies with a key named year
 and a sort key named title.");
        createTable(ddb, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Loading data into the Amazon DynamoDB table.");
        loadData(ddb, tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Getting data from the Movie table.");
        getItem(ddb);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Putting a record into the Amazon DynamoDB table.");
        putRecord(ddb);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Updating a record.");
        updateTableItem(ddb, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Scanning the Amazon DynamoDB table.");
```

```
        scanMovies(ddb, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Querying the Movies released in 2013.");
        queryTable(ddb);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        System.out.println(DASHES);

        ddb.close();
    }

    // Create a table with a Sort key.
    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE)
            .build();

        // Add KeySchemaElement objects to the list.
```

```
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)
            .billingMode(BillingMode.PAY_PER_REQUEST) //  DynamoDB automatically
scales based on traffic.
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            String newTable = response.tableDescription().tableName();
            System.out.println("The " + newTable + " was successfully created.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Query the table.
    public static void queryTable(DynamoDbClient ddb) {
        try {
            DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(ddb)
                .build();

            DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
            QueryConditional queryConditional = QueryConditional
                .keyEqualTo(Key.builder()
                    .partitionValue(2013)
                    .build());
```

```
            // Get items in the table and write out the ID value.
            Iterator<Movies> results =
custTable.query(queryConditional).items().iterator();
            String result = "";

            while (results.hasNext()) {
                Movies rec = results.next();
                System.out.println("The title of the movie is " + rec.getTitle());
                System.out.println("The movie information  is " + rec.getInfo());
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Scan the table.
    public static void scanMovies(DynamoDbClient ddb, String tableName) {
        System.out.println("******* Scanning all movies.\n");
        try {
            DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(ddb)
                .build();

            DynamoDbTable<Movies> custTable = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
            Iterator<Movies> results = custTable.scan().items().iterator();
            while (results.hasNext()) {
                Movies rec = results.next();
                System.out.println("The movie title is " + rec.getTitle());
                System.out.println("The movie year is " + rec.getYear());
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Load data into the table.
    public static void loadData(DynamoDbClient ddb, String tableName, String
fileName) throws IOException {
        DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
```

```
            .dynamoDbClient(ddb)
            .build();

        DynamoDbTable<Movies> mappedTable = enhancedClient.table("Movies",
 TableSchema.fromBean(Movies.class));
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
 ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0;
        while (iter.hasNext()) {
            // Only add 200 Movies to the table.
            if (t == 200)
                break;
            currentNode = (ObjectNode) iter.next();

            int year = currentNode.path("year").asInt();
            String title = currentNode.path("title").asText();
            String info = currentNode.path("info").toString();

            Movies movies = new Movies();
            movies.setYear(year);
            movies.setTitle(title);
            movies.setInfo(info);

            // Put the data into the Amazon DynamoDB Movie table.
            mappedTable.putItem(movies);
            t++;
        }
    }

    // Update the record to include show only directors.
    public static void updateTableItem(DynamoDbClient ddb, String tableName) {
        HashMap<String, AttributeValue> itemKey = new HashMap<>();
        itemKey.put("year", AttributeValue.builder().n("1933").build());
        itemKey.put("title", AttributeValue.builder().s("King Kong").build());

        HashMap<String, AttributeValueUpdate> updatedValues = new HashMap<>();
        updatedValues.put("info", AttributeValueUpdate.builder()
            .value(AttributeValue.builder().s("{\"directors\":[\"Merian C. Cooper\",
\"Ernest B. Schoedsack\"]")
                .build())
            .action(AttributeAction.PUT)
```

```
            .build());

        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(itemKey)
            .attributeUpdates(updatedValues)
            .build();

        try {
            ddb.updateItem(request);
        } catch (ResourceNotFoundException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }

        System.out.println("Item was updated!");
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
        DeleteTableRequest request = DeleteTableRequest.builder()
            .tableName(tableName)
            .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName + " was successfully deleted!");
    }

    public static void putRecord(DynamoDbClient ddb) {
        try {
            DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
                .dynamoDbClient(ddb)
                .build();

            DynamoDbTable<Movies> table = enhancedClient.table("Movies",
TableSchema.fromBean(Movies.class));
```

```
            // Populate the Table.
            Movies record = new Movies();
            record.setYear(2020);
            record.setTitle("My Movie2");
            record.setInfo("no info");
            table.putItem(record);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Added a new movie to the table.");
    }

    public static void getItem(DynamoDbClient ddb) {

        HashMap<String, AttributeValue> keyToGet = new HashMap<>();
        keyToGet.put("year", AttributeValue.builder()
            .n("1933")
            .build());

        keyToGet.put("title", AttributeValue.builder()
            .s("King Kong")
            .build());

        GetItemRequest request = GetItemRequest.builder()
            .key(keyToGet)
            .tableName("Movies")
            .build();

        try {
            Map<String, AttributeValue> returnedItem = ddb.getItem(request).item();

            if (returnedItem != null) {
                Set<String> keys = returnedItem.keySet();
                System.out.println("Amazon DynamoDB table attributes: \n");

                for (String key1 : keys) {
                    System.out.format("%s: %s\n", key1,
returnedItem.get(key1).toString());
                }
            } else {
                System.out.format("No item found with the key %s!\n", "year");
```

```
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Query
  - Scan
  - UpdateItem

操作

`BatchGetItem`

以下代码示例演示了如何使用 `BatchGetItem`。

适用于 Java 的 SDK 2.x

> **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。

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器获取批量项目。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考BatchGetItem中的。

`BatchWriteItem`

以下代码示例演示了如何使用 `BatchWriteItem`。

适用于 Java 的 SDK 2.x

> **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。

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(item1Attribut
```

```
        // Set item 2.
        Map<String, AttributeValue> item2Attributes = new HashMap<>();
        item2Attributes.put("Artist",
 AttributeValue.builder().s("Artist2").build());
        item2Attributes.put("Rating", AttributeValue.builder().s("4").build());
        item2Attributes.put("Comments", AttributeValue.builder().s("Nice
 melody.").build());
        item2Attributes.put("SongTitle",
 AttributeValue.builder().s("SongTitle2").build());

 writeRequests.add(WriteRequest.builder().putRequest(PutRequest.builder().item(item2Attribut

        try {
            // Create the BatchWriteItemRequest.
            BatchWriteItemRequest batchWriteItemRequest =
 BatchWriteItemRequest.builder()
                .requestItems(Map.of(tableName, writeRequests))
                .build();

            // Execute the BatchWriteItem operation.
            BatchWriteItemResponse batchWriteItemResponse =
 dynamoDbClient.batchWriteItem(batchWriteItemRequest);

            // Process the response.
            System.out.println("Batch write successful: " + batchWriteItemResponse);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考BatchWriteItem中的。

`CreateTable`

以下代码示例演示了如何使用 `CreateTable`。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.BillingMode;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.OnDemandThroughput;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateTable {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <tableName> <key>

            Where:
                tableName - The Amazon DynamoDB table to create (for example,
 Music3).
                key - The key for the Amazon DynamoDB table (for example, Artist).
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String tableName = args[0];
        String key = args[1];
        System.out.println("Creating an Amazon DynamoDB table " + tableName + " with
a simple primary key: " + key);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        String result = createTable(ddb, tableName, key);
        System.out.println("New table is " + result);
        ddb.close();
    }

    public static String createTable(DynamoDbClient ddb, String tableName, String
key) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        CreateTableRequest request = CreateTableRequest.builder()
            .attributeDefinitions(AttributeDefinition.builder()
                .attributeName(key)
                .attributeType(ScalarAttributeType.S)
                .build())
            .keySchema(KeySchemaElement.builder()
                .attributeName(key)
                .keyType(KeyType.HASH)
                .build())
            .billingMode(BillingMode.PAY_PER_REQUEST) //  DynamoDB automatically
scales based on traffic.
            .tableName(tableName)
            .build();

        String newTable;
        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            newTable = response.tableDescription().tableName();
```

```
            return newTable;

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateTable中的。

`DeleteItem`

以下代码示例演示了如何使用 `DeleteItem`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteItem中的。

`DeleteTable`

以下代码示例演示了如何使用 `DeleteTable`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteTable中的。

DescribeTable

以下代码示例演示了如何使用 DescribeTable。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
import
 software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughputDescription;
import software.amazon.awssdk.services.dynamodb.model.TableDescription;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeTable {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <tableName>

                Where:
                    tableName - The Amazon DynamoDB table to get information about
 (for example, Music3).
                """;

        if (args.length != 1) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String tableName = args[0];
        System.out.format("Getting description for %s\n\n", tableName);
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
                .region(region)
                .build();

        describeDymamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static void describeDymamoDBTable(DynamoDbClient ddb, String tableName) {
        DescribeTableRequest request = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

        try {
            TableDescription tableInfo = ddb.describeTable(request).table();
            if (tableInfo != null) {
                System.out.format("Table name  : %s\n", tableInfo.tableName());
                System.out.format("Table ARN   : %s\n", tableInfo.tableArn());
                System.out.format("Status      : %s\n", tableInfo.tableStatus());
                System.out.format("Item count  : %d\n", tableInfo.itemCount());
                System.out.format("Size (bytes): %d\n", tableInfo.tableSizeBytes());

                ProvisionedThroughputDescription throughputInfo =
tableInfo.provisionedThroughput();
                System.out.println("Throughput");
                System.out.format("  Read Capacity : %d\n",
throughputInfo.readCapacityUnits());
                System.out.format("  Write Capacity: %d\n",
throughputInfo.writeCapacityUnits());

                List<AttributeDefinition> attributes =
tableInfo.attributeDefinitions();
                System.out.println("Attributes");
                for (AttributeDefinition a : attributes) {
                    System.out.format("  %s (%s)\n", a.attributeName(),
a.attributeType());
                }
```

```
            }

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("\nDone!");
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeTable中的。

`DescribeTimeToLive`

以下代码示例演示了如何使用 `DescribeTimeToLive`。

适用于 Java 的 SDK 2.x

使用 AWS SDK for Java 2.x描述现有 DynamoDB 表上的 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.logging.Level;
import java.util.logging.Logger;

    public DescribeTimeToLiveResponse describeTTL(final String tableName, final
 Region region) {
        final DescribeTimeToLiveRequest request =
            DescribeTimeToLiveRequest.builder().tableName(tableName).build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            return ddb.describeTimeToLive(request);
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
```

```
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeTimeToLive中的。

`GetItem`

以下代码示例演示了如何使用 `GetItem`。

适用于 Java 的 SDK 2.x

> **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。

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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetItem中的。

`ListTables`

以下代码示例演示了如何使用 `ListTables`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ListTables](#)中的。

`PutItem`

以下代码示例演示了如何使用 `PutItem`。

适用于 Java 的 SDK 2.x

> **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。

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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutItem中的。

Query

以下代码示例演示了如何使用 Query。

适用于 Java 的 SDK 2.x

> **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。

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(expressionAttributesNames)
                    .expressionAttributeValues(expressionAttributeValues)
                    .build();

            System.out.println("=== Movie Titles ===");
            QueryResponse response = ddb.query(request);
            response.items()
                    .forEach(movie -> System.out.println(movie.get("title").s()));

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 Query。

Scan

以下代码示例演示了如何使用 Scan。

适用于 Java 的 SDK 2.x

> **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。

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《AWS SDK for Java 2.x API Reference》中的 Scan。

`UpdateItem`

以下代码示例演示了如何使用 `UpdateItem`。

适用于 Java 的 SDK 2.x

> **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。

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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考UpdateItem中的。

`UpdateTimeToLive`

以下代码示例演示了如何使用 `UpdateTimeToLive`。

适用于 Java 的 SDK 2.x

使用 AWS SDK for Java 2.x在现有 DynamoDB 表上启用 TTL。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.TimeToLiveSpecification;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveResponse;

import java.util.logging.Level;
import java.util.logging.Logger;

    public UpdateTimeToLiveResponse enableTTL(final String tableName, final String
 attributeName, final Region region) {
```

```
        final TimeToLiveSpecification ttlSpec = TimeToLiveSpecification.builder()
            .attributeName(attributeName)
            .enabled(true)
            .build();

        final UpdateTimeToLiveRequest request = UpdateTimeToLiveRequest.builder()
            .tableName(tableName)
            .timeToLiveSpecification(ttlSpec)
            .build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            return ddb.updateTimeToLive(request);
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
```

使用 AWS SDK for Java 2.x在现有 DynamoDB 表上禁用 TTL。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.TimeToLiveSpecification;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateTimeToLiveResponse;

import java.util.logging.Level;
import java.util.logging.Logger;

    public UpdateTimeToLiveResponse disableTTL(
        final String tableName, final String attributeName, final Region region) {
        final TimeToLiveSpecification ttlSpec = TimeToLiveSpecification.builder()
            .attributeName(attributeName)
            .enabled(false)
            .build();
```

```
        final UpdateTimeToLiveRequest request = UpdateTimeToLiveRequest.builder()
            .tableName(tableName)
            .timeToLiveSpecification(ttlSpec)
            .build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            return ddb.updateTimeToLive(request);
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考UpdateTimeToLive中的。

场景

构建应用程序以将数据提交到 DynamoDB 表

以下代码示例演示如何构建将数据提交到 Amazon DynamoDB 表并在用户更新该表时通知您的应用程序。

适用于 Java 的 SDK 2.x

展示如何创建动态 Web 应用程序，该应用程序使用 Amazon DynamoDB Java API 提交数据并使用 Amazon Simple Notification Service Java API 发送文本消息。

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。

本示例中使用的服务

- DynamoDB
- Amazon SNS

将多个值与单个属性进行比较

以下代码示例演示如何在 DynamoDB 中将多个值与单个属性进行比较。

- 使用 IN 运算符将多个值与单个属性进行比较。
- 将 IN 运算符与多个 OR 条件进行比较。
- 了解使用 IN 对性能和表达式复杂性的好处。

适用于 Java 的 SDK 2.x

使用在 DynamoDB 中将多个值与单个属性进行比较。 AWS SDK for Java 2.x

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ScanRequest;
import software.amazon.awssdk.services.dynamodb.model.ScanResponse;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Locale;
import java.util.Map;

    /**
     * Queries a table using the IN operator to compare multiple values with a
 single attribute.
     *
     * <p>This method demonstrates how to use the IN operator in a filter expression
     * to match an attribute against multiple values.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param partitionKeyName The name of the partition key attribute
     * @param partitionKeyValue The value of the partition key to query
     * @param attributeName The name of the attribute to compare
     * @param valuesList List of values to compare against
     * @return The query response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
```

```
    public static QueryResponse compareMultipleValues(
        DynamoDbClient dynamoDbClient,
        String tableName,
        String partitionKeyName,
        AttributeValue partitionKeyValue,
        String attributeName,
        List<AttributeValue> valuesList) {

        // Create expression attribute names
        Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#pkName", partitionKeyName);
        expressionAttributeNames.put("#attrName", attributeName);

        // Create expression attribute values
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
        expressionAttributeValues.put(":pkValue", partitionKeyValue);

        // Add values for IN operator
        for (int i = 0; i < valuesList.size(); i++) {
            expressionAttributeValues.put(":val" + i, valuesList.get(i));
        }

        // Build the IN clause
        StringBuilder inClause = new StringBuilder();
        for (int i = 0; i < valuesList.size(); i++) {
            if (i > 0) {
                inClause.append(", ");
            }
            inClause.append(":val").append(i);
        }

        // Define the query parameters
        QueryRequest request = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression("#pkName = :pkValue")
            .filterExpression("#attrName IN (" + inClause.toString() + ")")
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the query operation
        return dynamoDbClient.query(request);
    }
```

```
    /**
     * Queries a table using multiple OR conditions to compare multiple values with
a single attribute.
     *
     * <p>This method demonstrates the alternative approach to using the IN
operator,
     * by using multiple OR conditions.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param partitionKeyName The name of the partition key attribute
     * @param partitionKeyValue The value of the partition key to query
     * @param attributeName The name of the attribute to compare
     * @param valuesList List of values to compare against
     * @return The query response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static QueryResponse compareWithOrConditions(
        DynamoDbClient dynamoDbClient,
        String tableName,
        String partitionKeyName,
        AttributeValue partitionKeyValue,
        String attributeName,
        List<AttributeValue> valuesList) {

        // Create expression attribute names
        Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#pkName", partitionKeyName);
        expressionAttributeNames.put("#attrName", attributeName);

        // Create expression attribute values
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
        expressionAttributeValues.put(":pkValue", partitionKeyValue);

        // Add values for OR conditions
        for (int i = 0; i < valuesList.size(); i++) {
            expressionAttributeValues.put(":val" + i, valuesList.get(i));
        }

        // Build the OR conditions
        StringBuilder orConditions = new StringBuilder();
        for (int i = 0; i < valuesList.size(); i++) {
            if (i > 0) {
                orConditions.append(" OR ");
```

```
            }
            orConditions.append("#attrName = :val").append(i);
        }

        // Define the query parameters
        QueryRequest request = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression("#pkName = :pkValue")
            .filterExpression(orConditions.toString())
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the query operation
        return dynamoDbClient.query(request);
    }

    /**
     * Compares the performance of using the IN operator versus multiple OR
conditions.
     *
     * <p>This method demonstrates the performance difference between using the IN
operator
     * and using multiple OR conditions.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param partitionKeyName The name of the partition key attribute
     * @param partitionKeyValue The value of the partition key to query
     * @param attributeName The name of the attribute to compare
     * @param valuesList List of values to compare against
     * @return Map containing the performance comparison results
     */
    public static Map<String, Object> comparePerformance(
        DynamoDbClient dynamoDbClient,
        String tableName,
        String partitionKeyName,
        AttributeValue partitionKeyValue,
        String attributeName,
        List<AttributeValue> valuesList) {

        Map<String, Object> results = new HashMap<>();

        try {
```

```
            // Measure performance of IN operator
            long inStartTime = System.nanoTime();
            QueryResponse inResponse = compareMultipleValues(
                dynamoDbClient, tableName, partitionKeyName, partitionKeyValue,
attributeName, valuesList);
            long inEndTime = System.nanoTime();
            long inDuration = inEndTime - inStartTime;

            // Measure performance of OR conditions
            long orStartTime = System.nanoTime();
            QueryResponse orResponse = compareWithOrConditions(
                dynamoDbClient, tableName, partitionKeyName, partitionKeyValue,
attributeName, valuesList);
            long orEndTime = System.nanoTime();
            long orDuration = orEndTime - orStartTime;

            // Record results
            results.put("inOperatorDuration", inDuration);
            results.put("orConditionsDuration", orDuration);
            results.put("inOperatorItems", inResponse.count());
            results.put("orConditionsItems", orResponse.count());
            results.put("inOperatorExpression", "IN operator with " +
valuesList.size() + " values");
            results.put("orConditionsExpression", valuesList.size() + " OR
conditions");
            results.put("success", true);

        } catch (DynamoDbException e) {
            results.put("success", false);
            results.put("error", e.getMessage());
        }

        return results;
    }

    /**
     * Scans a table using the IN operator with a large number of values.
     *
     * <p>This method demonstrates how to use the IN operator with a large number of
values,
     * which can help stay within the 300 operator limit.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
```

```
     * @param attributeName The name of the attribute to compare
     * @param valuesList List of values to compare against
     * @return The scan response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static ScanResponse scanWithLargeInClause(
        DynamoDbClient dynamoDbClient, String tableName, String attributeName,
List<AttributeValue> valuesList) {

        // Create expression attribute names
        Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#attrName", attributeName);

        // Create expression attribute values
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();

        // Add values for IN operator
        for (int i = 0; i < valuesList.size(); i++) {
            expressionAttributeValues.put(":val" + i, valuesList.get(i));
        }

        // Build the IN clause
        StringBuilder inClause = new StringBuilder();
        for (int i = 0; i < valuesList.size(); i++) {
            if (i > 0) {
                inClause.append(", ");
            }
            inClause.append(":val").append(i);
        }

        // Define the scan parameters
        ScanRequest request = ScanRequest.builder()
            .tableName(tableName)
            .filterExpression("#attrName IN (" + inClause.toString() + ")")
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the scan operation
        return dynamoDbClient.scan(request);
    }

    /**
     * Generates a list of sample values for testing.
```

```
     *
     * <p>Helper method to generate a list of sample values for testing.
     *
     * @param valueType The type of values to generate (string, number, or boolean)
     * @param count The number of values to generate
     * @return List of generated attribute values
     */
    public static List<AttributeValue> generateSampleValues(String valueType, int
count) {
        List<AttributeValue> values = new ArrayList<>();

        for (int i = 0; i < count; i++) {
            AttributeValue value;

            switch (valueType.toLowerCase(Locale.ROOT)) {
                case "string":
                    value = AttributeValue.builder().s("Value" + i).build();
                    break;
                case "number":
                    value = AttributeValue.builder().n(String.valueOf(i)).build();
                    break;
                case "boolean":
                    value = AttributeValue.builder().bool(i % 2 == 0).build();
                    break;
                default:
                    throw new IllegalArgumentException("Unsupported value type: " +
valueType);
            }

            values.add(value);
        }

        return values;
    }
```

将多个值与进行比较的用法示例 AWS SDK for Java 2.x。

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        System.out.println("Demonstrating how to compare multiple values with a
single attribute in DynamoDB");
```

```
        try {
            // Example 1: Using the IN operator
            System.out.println("\nExample 1: Using the IN operator");
            List<AttributeValue> categories = List.of(
                AttributeValue.builder().s("Electronics").build(),
                AttributeValue.builder().s("Computers").build(),
                AttributeValue.builder().s("Accessories").build());

            QueryResponse inResponse = compareMultipleValues(
                dynamoDbClient,
                tableName,
                "Department",
                AttributeValue.builder().s("Retail").build(),
                "Category",
                categories);

            System.out.println("Found " + inResponse.count() + " items using IN
operator");
            System.out.println("Items: " + inResponse.items());

            // Example 2: Using multiple OR conditions
            System.out.println("\nExample 2: Using multiple OR conditions");
            QueryResponse orResponse = compareWithOrConditions(
                dynamoDbClient,
                tableName,
                "Department",
                AttributeValue.builder().s("Retail").build(),
                "Category",
                categories);

            System.out.println("Found " + orResponse.count() + " items using OR
conditions");
            System.out.println("Items: " + orResponse.items());

            // Example 3: Performance comparison
            System.out.println("\nExample 3: Performance comparison");
            Map<String, Object> perfComparison = comparePerformance(
                dynamoDbClient,
                tableName,
                "Department",
                AttributeValue.builder().s("Retail").build(),
                "Category",
                categories);
```

```
            if ((boolean) perfComparison.get("success")) {
                System.out.println("IN operator duration: " +
perfComparison.get("inOperatorDuration") + " ns");
                System.out.println("OR conditions duration: " +
perfComparison.get("orConditionsDuration") + " ns");
                System.out.println("IN operator found " +
perfComparison.get("inOperatorItems") + " items");
                System.out.println("OR conditions found " +
perfComparison.get("orConditionsItems") + " items");
                System.out.println("Expression complexity comparison:");
                System.out.println("  IN operator: " +
perfComparison.get("inOperatorExpression"));
                System.out.println("  OR conditions: " +
perfComparison.get("orConditionsExpression"));
            } else {
                System.out.println("Performance comparison failed: " +
perfComparison.get("error"));
            }

            // Example 4: Using IN with a large number of values
            System.out.println("\nExample 4: Using IN with a large number of
values");
            List<AttributeValue> productIds = generateSampleValues("string", 20);

            ScanResponse largeInResponse = scanWithLargeInClause(dynamoDbClient,
tableName, "ProductId", productIds);

            System.out.println(
                "Found " + largeInResponse.count() + " items using IN with " +
productIds.size() + " values");

            // Explain the benefits of using IN
            System.out.println("\nKey points about using the IN operator in
DynamoDB:");
            System.out.println("1. The IN operator allows comparing a single
attribute against multiple values");
            System.out.println("2. IN is more concise than using multiple OR
conditions");
            System.out.println("3. IN counts as only 1 operator regardless of the
number of values");
            System.out.println("4. Multiple OR conditions count as 1 operator per
condition plus 1 per OR");
            System.out.println("5. Using IN helps stay within the 300 operator limit
for complex expressions");
```

```
            System.out.println("6. IN can be used in filter expressions and
 condition expressions");
            System.out.println("7. The IN operator supports up to 100 comparison
 values");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - Query
  - Scan

有条件地更新项目的 TTL

以下代码示例演示如何有条件地更新项目的 TTL。

适用于 Java 的 SDK 2.x

使用条件更新表中现有 DynamoDB 项目的 TTL。

```
package com.amazon.samplelib.ttl;

import com.amazon.samplelib.CodeSampleUtils;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import
 software.amazon.awssdk.services.dynamodb.model.ConditionalCheckFailedException;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.Map;
import java.util.Optional;

/**
```

```
 * Updates an item in a DynamoDB table with TTL attributes using a conditional
 expression.
 * This class demonstrates how to conditionally update TTL expiration timestamps.
 */
public class UpdateTTLConditional {

    private static final String USAGE =
        """
            Usage:
                <tableName> <primaryKey> <sortKey> <region>
            Where:
                tableName - The Amazon DynamoDB table being queried.
                primaryKey - The name of the primary key. Also known as the hash or
 partition key.
                sortKey - The name of the sort key. Also known as the range
 attribute.
                region (optional) - The AWS region that the Amazon DynamoDB table is
 located in. (Default: us-east-1)
            """;
    private static final int DAYS_TO_EXPIRE = 90;
    private static final int SECONDS_PER_DAY = 24 * 60 * 60;
    private static final String PRIMARY_KEY_ATTR = "primaryKey";
    private static final String SORT_KEY_ATTR = "sortKey";
    private static final String UPDATED_AT_ATTR = "updatedAt";
    private static final String EXPIRE_AT_ATTR = "expireAt";
    private static final String UPDATE_EXPRESSION = "SET " + UPDATED_AT_ATTR + "=:c,
" + EXPIRE_AT_ATTR + "=:e";
    private static final String CONDITION_EXPRESSION = "attribute_exists(" +
 PRIMARY_KEY_ATTR + ")";
    private static final String SUCCESS_MESSAGE = "%s UpdateItem operation with TTL
 successful.";
    private static final String CONDITION_FAILED_MESSAGE = "Condition check failed.
 Item does not exist.";
    private static final String TABLE_NOT_FOUND_ERROR = "Error: The Amazon DynamoDB
 table \"%s\" can't be found.";

    private final DynamoDbClient dynamoDbClient;

    /**
     * Constructs an UpdateTTLConditional with a default DynamoDB client.
     */
    public UpdateTTLConditional() {
        this.dynamoDbClient = null;
    }
```

```
    /**
     * Constructs an UpdateTTLConditional with the specified DynamoDB client.
     *
     * @param dynamoDbClient The DynamoDB client to use
     */
    public UpdateTTLConditional(final DynamoDbClient dynamoDbClient) {
        this.dynamoDbClient = dynamoDbClient;
    }

    /**
     * Main method to demonstrate conditionally updating an item with TTL.
     *
     * @param args Command line arguments
     */
    public static void main(final String[] args) {
        try {
            int result = new UpdateTTLConditional().processArgs(args);
            System.exit(result);
        } catch (Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Process command line arguments and conditionally update an item with TTL.
     *
     * @param args Command line arguments
     * @return 0 if successful, non-zero otherwise
     * @throws ResourceNotFoundException If the table doesn't exist
     * @throws DynamoDbException If an error occurs during the operation
     * @throws IllegalArgumentException If arguments are invalid
     */
    public int processArgs(final String[] args) {
        // Argument validation (remove or replace this line when reusing this code)
        CodeSampleUtils.validateArgs(args, new int[] {3, 4}, USAGE);

        final String tableName = args[0];
        final String primaryKey = args[1];
        final String sortKey = args[2];
        final Region region = Optional.ofNullable(args.length > 3 ? args[3] : null)
            .map(Region::of)
            .orElse(Region.US_EAST_1);
```

```
        // Get current time in epoch second format
        final long currentTime = System.currentTimeMillis() / 1000;

        // Calculate expiration time 90 days from now in epoch second format
        final long expireDate = currentTime + (DAYS_TO_EXPIRE * SECONDS_PER_DAY);

        // Create the key map for the item to update
        final Map<String, AttributeValue> keyMap = Map.of(
            PRIMARY_KEY_ATTR, AttributeValue.builder().s(primaryKey).build(),
            SORT_KEY_ATTR, AttributeValue.builder().s(sortKey).build());

        // Create the expression attribute values
        final Map<String, AttributeValue> expressionAttributeValues = Map.of(
            ":c", AttributeValue.builder().n(String.valueOf(currentTime)).build(),
            ":e", AttributeValue.builder().n(String.valueOf(expireDate)).build());

        final UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(keyMap)
            .updateExpression(UPDATE_EXPRESSION)
            .conditionExpression(CONDITION_EXPRESSION)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            final UpdateItemResponse response = ddb.updateItem(request);
            System.out.println(String.format(SUCCESS_MESSAGE, tableName));
            return 0;
        } catch (ConditionalCheckFailedException e) {
            System.err.println(CONDITION_FAILED_MESSAGE);
            throw e;
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考UpdateItem中的。

对表达式运算符进行计数

以下代码示例演示如何在 DynamoDB 中对表达式运算符进行计数。

- 了解 DynamoDB 的 300 个运算符限制。
- 对复杂表达式中的运算符进行计数。
- 优化表达式以保持在限制范围内。

适用于 Java 的 SDK 2.x

演示使用 AWS SDK for Java 2.x对表达式运算符进行计数。

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.Map;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

    /**
     * Creates a complex filter expression with a specified number of conditions.
     *
     * <p>This method demonstrates how to generate a complex expression with
     * a specific number of operators to test the 300 operator limit.
     *
     * @param conditionsCount Number of conditions to include
     * @param useAnd Whether to use AND (true) or OR (false) between conditions
     * @return Map containing the filter expression, attribute values, and operator
 count
     */
    public static Map<String, Object> createComplexFilterExpression(int
 conditionsCount, boolean useAnd) {
        // Initialize the expression parts and attribute values
```

```
        StringBuilder filterExpression = new StringBuilder();
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();

        // Generate the specified number of conditions
        for (int i = 0; i < conditionsCount; i++) {
            // Add the operator between conditions (except for the first one)
            if (i > 0) {
                filterExpression.append(useAnd ? " AND " : " OR ");
            }

            // Alternate between different comparison operators for variety
            String valueKey = ":val" + i;

            switch (i % 5) {
                case 0:
                    filterExpression.append("attribute").append(i).append(" =
").append(valueKey);
                    expressionAttributeValues.put(
                        valueKey, AttributeValue.builder().s("value" + i).build());
                    break;
                case 1:
                    filterExpression.append("attribute").append(i).append(" >
").append(valueKey);
                    expressionAttributeValues.put(
                        valueKey,
AttributeValue.builder().n(String.valueOf(i)).build());
                    break;
                case 2:
                    filterExpression.append("attribute").append(i).append(" <
").append(valueKey);
                    expressionAttributeValues.put(
                        valueKey,
                        AttributeValue.builder().n(String.valueOf(i * 10)).build());
                    break;
                case 3:
                    filterExpression
                        .append("contains(attribute")
                        .append(i)
                        .append(", ")
                        .append(valueKey)
                        .append(")");
                    expressionAttributeValues.put(
                        valueKey, AttributeValue.builder().s("substring" +
i).build());
```

```
                    break;
                case 4:
                    filterExpression
                        .append("attribute_exists(attribute")
                        .append(i)
                        .append(")");
                    break;
                default:
                    // This case will never be reached, but added to satisfy
checkstyle
                    break;
            }
        }

        // Calculate the operator count
        // Each condition has 1 operator (=, >, <, contains, attribute_exists)
        // Each AND or OR between conditions is 1 operator
        int operatorCount = conditionsCount + (conditionsCount > 0 ? conditionsCount
- 1 : 0);

        // Create the result map
        Map<String, Object> result = new HashMap<>();
        result.put("filterExpression", filterExpression.toString());
        result.put("expressionAttributeValues", expressionAttributeValues);
        result.put("operatorCount", operatorCount);

        return result;
    }

    /**
     * Creates a complex update expression with a specified number of operations.
     *
     * <p>This method demonstrates how to generate a complex update expression with
     * a specific number of operators to test the 300 operator limit.
     *
     * @param operationsCount Number of operations to include
     * @return Map containing the update expression, attribute values, and operator
count
     */
    public static Map<String, Object> createComplexUpdateExpression(int
operationsCount) {
        // Initialize the expression parts and attribute values
        StringBuilder updateExpression = new StringBuilder("SET ");
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
```

```
        // Generate the specified number of SET operations
        for (int i = 0; i < operationsCount; i++) {
            // Add comma between operations (except for the first one)
            if (i > 0) {
                updateExpression.append(", ");
            }

            // Alternate between different types of SET operations
            String valueKey = ":val" + i;

            switch (i % 3) {
                case 0:
                    // Simple assignment (1 operator: =)
                    updateExpression.append("attribute").append(i).append(" =
").append(valueKey);
                    expressionAttributeValues.put(
                        valueKey, AttributeValue.builder().s("value" + i).build());
                    break;
                case 1:
                    // Addition (2 operators: = and +)
                    updateExpression
                        .append("attribute")
                        .append(i)
                        .append(" = attribute")
                        .append(i)
                        .append(" + ")
                        .append(valueKey);
                    expressionAttributeValues.put(
                        valueKey,
AttributeValue.builder().n(String.valueOf(i)).build());
                    break;
                case 2:
                    // Conditional assignment with if_not_exists (2 operators: = and
if_not_exists)
                    updateExpression
                        .append("attribute")
                        .append(i)
                        .append(" = if_not_exists(attribute")
                        .append(i)
                        .append(", ")
                        .append(valueKey)
                        .append(")");
                    expressionAttributeValues.put(
```

```
                        valueKey,
                        AttributeValue.builder().n(String.valueOf(i * 10)).build());
                    break;
                default:
                    // This case will never be reached, but added to satisfy
checkstyle
                    break;
            }
        }

        // Calculate the operator count
        // Each operation has 1-2 operators as noted above
        int operatorCount = 0;
        for (int i = 0; i < operationsCount; i++) {
            operatorCount += (i % 3 == 0) ? 1 : 2;
        }

        // Create the result map
        Map<String, Object> result = new HashMap<>();
        result.put("updateExpression", updateExpression.toString());
        result.put("expressionAttributeValues", expressionAttributeValues);
        result.put("operatorCount", operatorCount);

        return result;
    }

    /**
     * Test the operator limit by attempting an operation with a complex expression.
     *
     * <p>This method demonstrates what happens when an expression approaches or
     * exceeds the 300 operator limit.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param operatorCount Target number of operators to include
     * @return Map containing the result of the operation attempt
     */
    public static Map<String, Object> testOperatorLimit(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, int operatorCount) {

        // Create a complex update expression with the specified operator count
        Map<String, Object> expressionData =
```

```
            createComplexUpdateExpression((int) Math.ceil(operatorCount / 1.5)); //
Adjust to get close to target count

        String updateExpression = (String) expressionData.get("updateExpression");
        @SuppressWarnings("unchecked")
        Map<String, AttributeValue> expressionAttributeValues =
            (Map<String, AttributeValue>)
expressionData.get("expressionAttributeValues");
        int actualCount = (int) expressionData.get("operatorCount");

        System.out.println("Generated update expression with approximately " +
actualCount + " operators");

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression(updateExpression)
            .expressionAttributeValues(expressionAttributeValues)
            .returnValues("UPDATED_NEW")
            .build();

        try {
            // Attempt the update operation
            UpdateItemResponse response = dynamoDbClient.updateItem(request);

            Map<String, Object> result = new HashMap<>();
            result.put("success", true);
            result.put("message", "Operation succeeded with " + actualCount + "
operators");
            result.put("data", response);
            return result;

        } catch (DynamoDbException e) {
            // Check if the error is due to exceeding the operator limit
            if (e.getMessage().contains("too many operators")) {
                Map<String, Object> result = new HashMap<>();
                result.put("success", false);
                result.put("message", "Operation failed: " + e.getMessage());
                result.put("operatorCount", actualCount);
                return result;
            }

            // Return other errors
```

```
            Map<String, Object> result = new HashMap<>();
            result.put("success", false);
            result.put("message", "Operation failed: " + e.getMessage());
            result.put("error", e);
            return result;
        }
    }

    /**
     * Break down a complex expression into multiple simpler operations.
     *
     * <p>This method demonstrates how to handle expressions that would exceed
     * the 300 operator limit by breaking them into multiple operations.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param totalOperations Total number of operations to perform
     * @return Map containing the results of the operations
     */
    public static Map<String, Object> breakDownComplexExpression(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, int totalOperations) {

        // Calculate how many operations we can safely include in each batch
        // Using 150 as a conservative limit (well below 300)
        final int operationsPerBatch = 100;
        final int batchCount = (int) Math.ceil((double) totalOperations /
operationsPerBatch);

        System.out.println("Breaking down " + totalOperations + " operations into "
+ batchCount + " batches");

        Map<String, Object> results = new HashMap<>();
        results.put("totalBatches", batchCount);

        Map<Integer, Map<String, Object>> batchResults = new HashMap<>();

        // Process each batch
        for (int batch = 0; batch < batchCount; batch++) {
            // Calculate the operations for this batch
            int batchStart = batch * operationsPerBatch;
            int batchEnd = Math.min(batchStart + operationsPerBatch,
totalOperations);
```

```
            int batchSize = batchEnd - batchStart;

            System.out.println(
                "Processing batch " + (batch + 1) + "/" + batchCount + " with " +
batchSize + " operations");

            // Create an update expression for this batch
            Map<String, Object> expressionData =
createComplexUpdateExpression(batchSize);

            String updateExpression = (String)
expressionData.get("updateExpression");
            @SuppressWarnings("unchecked")
            Map<String, AttributeValue> expressionAttributeValues =
                (Map<String, AttributeValue>)
expressionData.get("expressionAttributeValues");
            int operatorCount = (int) expressionData.get("operatorCount");

            // Define the update parameters
            UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression(updateExpression)
                .expressionAttributeValues(expressionAttributeValues)
                .returnValues("UPDATED_NEW")
                .build();

            try {
                // Perform the update operation for this batch
                UpdateItemResponse response = dynamoDbClient.updateItem(request);

                Map<String, Object> batchResult = new HashMap<>();
                batchResult.put("batch", batch + 1);
                batchResult.put("success", true);
                batchResult.put("operatorCount", operatorCount);
                batchResult.put("attributes", response.attributes());

                batchResults.put(batch, batchResult);

            } catch (DynamoDbException e) {
                Map<String, Object> batchResult = new HashMap<>();
                batchResult.put("batch", batch + 1);
                batchResult.put("success", false);
                batchResult.put("operatorCount", operatorCount);
```

```
                batchResult.put("error", e.getMessage());

                batchResults.put(batch, batchResult);

                // Continue with next batch instead of breaking
                continue;
            }
        }

        results.put("results", batchResults);
        return results;
    }

    /**
     * Count operators in a DynamoDB expression based on the rules in the
documentation.
     *
     * <p>This method demonstrates how operators are counted according to the
     * DynamoDB documentation.
     *
     * @param expression The DynamoDB expression to analyze
     * @return Map containing the breakdown of operator counts
     */
    public static Map<String, Integer> countOperatorsInExpression(String expression)
{
        // Initialize counters for different operator types
        Map<String, Integer> counts = new HashMap<>();
        counts.put("comparisonOperators", 0);
        counts.put("logicalOperators", 0);
        counts.put("functions", 0);
        counts.put("arithmeticOperators", 0);
        counts.put("specialOperators", 0);
        counts.put("total", 0);

        // Count comparison operators (=, <>, <, <=, >, >=)
        // This is a simplified approach and may not catch all cases
        int comparisonCount = 0;
        Pattern comparisonPattern = Pattern.compile("(=|<>|<=|>=|<|>)");
        Matcher comparisonMatcher = comparisonPattern.matcher(expression);
        while (comparisonMatcher.find()) {
            comparisonCount++;
        }
        counts.put("comparisonOperators", comparisonCount);
```

```
        // Count logical operators (AND, OR, NOT)
        int andCount = countOccurrences(expression, "\\bAND\\b");
        int orCount = countOccurrences(expression, "\\bOR\\b");
        int notCount = countOccurrences(expression, "\\bNOT\\b");
        counts.put("logicalOperators", andCount + orCount + notCount);

        // Count functions (attribute_exists, attribute_not_exists, attribute_type,
 begins_with, contains, size)
        int functionCount = countOccurrences(
            expression,
            "\\b(attribute_exists|attribute_not_exists|attribute_type|begins_with|
contains|size|if_not_exists)\\(");
        counts.put("functions", functionCount);

        // Count arithmetic operators (+ and -)
        // This is a simplified approach and may not catch all cases
        int arithmeticCount = 0;
        Pattern arithmeticPattern = Pattern.compile("[a-zA-Z0-9_)\\]]\\s*[\\+\\-]\
\s*[a-zA-Z0-9_:(]");
        Matcher arithmeticMatcher = arithmeticPattern.matcher(expression);
        while (arithmeticMatcher.find()) {
            arithmeticCount++;
        }
        counts.put("arithmeticOperators", arithmeticCount);

        // Count special operators (BETWEEN, IN)
        int betweenCount = countOccurrences(expression, "\\bBETWEEN\\b");
        int inCount = countOccurrences(expression, "\\bIN\\b");
        counts.put("specialOperators", betweenCount + inCount);

        // Add extra operators for BETWEEN (each BETWEEN includes an AND)
        int currentLogicalOps = counts.getOrDefault("logicalOperators", 0);
        counts.put("logicalOperators", currentLogicalOps + betweenCount);

        // Calculate total
        int total = counts.getOrDefault("comparisonOperators", 0)
            + counts.getOrDefault("logicalOperators", 0)
            + counts.getOrDefault("functions", 0)
            + counts.getOrDefault("arithmeticOperators", 0)
            + counts.getOrDefault("specialOperators", 0);
        counts.put("total", total);

        return counts;
    }
```

```
    /**
     * Helper method to count occurrences of a pattern in a string.
     *
     * @param text The text to search in
     * @param regex The regular expression pattern to search for
     * @return The number of occurrences
     */
    private static int countOccurrences(String text, String regex) {
        final Pattern pattern = Pattern.compile(regex);
        final Matcher matcher = pattern.matcher(text);
        int count = 0;
        while (matcher.find()) {
            count++;
        }
        return count;
    }
```

使用表达式运算符计数的用法示例 AWS SDK for Java 2.x。

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating DynamoDB expression operator counting and
the 300 operator limit");

        try {
            // Example 1: Analyze a simple expression
            System.out.println("\nExample 1: Analyzing a simple expression");
            String simpleExpression = "Price = :price AND Rating > :rating AND
Category IN (:cat1, :cat2, :cat3)";
            Map<String, Integer> simpleCount =
countOperatorsInExpression(simpleExpression);

            System.out.println("Expression: " + simpleExpression);
            System.out.println("Operator count breakdown:");
            System.out.println("- Comparison operators: " +
simpleCount.get("comparisonOperators"));
```

```
            System.out.println("- Logical operators: " +
simpleCount.get("logicalOperators"));
            System.out.println("- Functions: " + simpleCount.get("functions"));
            System.out.println("- Arithmetic operators: " +
simpleCount.get("arithmeticOperators"));
            System.out.println("- Special operators: " +
simpleCount.get("specialOperators"));
            System.out.println("- Total operators: " + simpleCount.get("total"));

            // Example 2: Analyze a complex expression
            System.out.println("\nExample 2: Analyzing a complex expression");
            String complexExpression = "(attribute_exists(Category) AND Size
BETWEEN :min AND :max) OR "
                + "(Price > :price AND contains(Description, :keyword) AND "
                + "(Rating >= :minRating OR Reviews > :minReviews))";
            Map<String, Integer> complexCount =
countOperatorsInExpression(complexExpression);

            System.out.println("Expression: " + complexExpression);
            System.out.println("Operator count breakdown:");
            System.out.println("- Comparison operators: " +
complexCount.get("comparisonOperators"));
            System.out.println("- Logical operators: " +
complexCount.get("logicalOperators"));
            System.out.println("- Functions: " + complexCount.get("functions"));
            System.out.println("- Arithmetic operators: " +
complexCount.get("arithmeticOperators"));
            System.out.println("- Special operators: " +
complexCount.get("specialOperators"));
            System.out.println("- Total operators: " + complexCount.get("total"));

            // Example 3: Test approaching the operator limit
            System.out.println("\nExample 3: Testing an expression approaching the
operator limit");
            Map<String, Object> approachingLimit = testOperatorLimit(dynamoDbClient,
tableName, key, 290);
            System.out.println(approachingLimit.get("message"));

            // Example 4: Test exceeding the operator limit
            System.out.println("\nExample 4: Testing an expression exceeding the
operator limit");
            Map<String, Object> exceedingLimit = testOperatorLimit(dynamoDbClient,
tableName, key, 310);
            System.out.println(exceedingLimit.get("message"));
```

```
            // Example 5: Breaking down a complex expression
            System.out.println("\nExample 5: Breaking down a complex expression into
multiple operations");
            Map<String, Object> breakdownResult =
breakDownComplexExpression(dynamoDbClient, tableName, key, 500);
            @SuppressWarnings("unchecked")
            Map<Integer, Map<String, Object>> results =
                (Map<Integer, Map<String, Object>>) breakdownResult.get("results");
            System.out.println(
                "Processed " + results.size() + " of " +
breakdownResult.get("totalBatches") + " batches");

            // Explain the operator counting rules
            System.out.println("\nKey points about DynamoDB expression operator
counting:");
            System.out.println("1. The maximum number of operators in any expression
is 300");
            System.out.println("2. Each comparison operator (=, <>, <, <=, >, >=)
counts as 1 operator");
            System.out.println("3. Each logical operator (AND, OR, NOT) counts as 1
operator");
            System.out.println("4. Each function call (attribute_exists, contains,
etc.) counts as 1 operator");
            System.out.println("5. Each arithmetic operator (+ or -) counts as 1
operator");
            System.out.println("6. BETWEEN counts as 2 operators (BETWEEN itself and
the AND within it)");
            System.out.println("7. IN counts as 1 operator regardless of the number
of values");
            System.out.println("8. Parentheses for grouping and attribute paths
don't count as operators");
            System.out.println("9. When you exceed the limit, the error always
reports '301 operators'");
            System.out.println("10. For complex operations, break them into multiple
smaller operations");

        } catch (Exception e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考UpdateItem中的。

创建无服务器应用程序来管理照片

以下代码示例演示如何创建无服务器应用程序，让用户能够使用标签管理照片。

适用于 Java 的 SDK 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 Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

创建带有全局二级索引的表

以下代码示例演示如何创建带有全局二级索引的表。

适用于 Java 的 SDK 2.x

使用创建带有全局二级索引的 DynamoDB 表。 AWS SDK for Java 2.x

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DeleteTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableRequest;
```

```
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GlobalSecondaryIndex;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.Projection;
import software.amazon.awssdk.services.dynamodb.model.ProjectionType;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
import software.amazon.awssdk.services.dynamodb.waiters.DynamoDbWaiter;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    public void createTable() {
        try {
            // Attribute definitions
            final List<AttributeDefinition> attributeDefinitions = new
 ArrayList<>();
            attributeDefinitions.add(AttributeDefinition.builder()
                .attributeName(ISSUE_ID_ATTR)
                .attributeType(ScalarAttributeType.S)
                .build());
            attributeDefinitions.add(AttributeDefinition.builder()
                .attributeName(TITLE_ATTR)
                .attributeType(ScalarAttributeType.S)
                .build());
            attributeDefinitions.add(AttributeDefinition.builder()
                .attributeName(CREATE_DATE_ATTR)
                .attributeType(ScalarAttributeType.S)
                .build());
            attributeDefinitions.add(AttributeDefinition.builder()
                .attributeName(DUE_DATE_ATTR)
                .attributeType(ScalarAttributeType.S)
                .build());

            // Key schema for table
            final List<KeySchemaElement> tableKeySchema = new ArrayList<>();
            tableKeySchema.add(KeySchemaElement.builder()
```

```
                .attributeName(ISSUE_ID_ATTR)
                .keyType(KeyType.HASH)
                .build()); // Partition key
            tableKeySchema.add(KeySchemaElement.builder()
                .attributeName(TITLE_ATTR)
                .keyType(KeyType.RANGE)
                .build()); // Sort key

            // Initial provisioned throughput settings for the indexes
            final ProvisionedThroughput ptIndex = ProvisionedThroughput.builder()
                .readCapacityUnits(1L)
                .writeCapacityUnits(1L)
                .build();

            // CreateDateIndex
            final List<KeySchemaElement> createDateKeySchema = new ArrayList<>();
            createDateKeySchema.add(KeySchemaElement.builder()
                .attributeName(CREATE_DATE_ATTR)
                .keyType(KeyType.HASH)
                .build());
            createDateKeySchema.add(KeySchemaElement.builder()
                .attributeName(ISSUE_ID_ATTR)
                .keyType(KeyType.RANGE)
                .build());

            final Projection createDateProjection = Projection.builder()
                .projectionType(ProjectionType.INCLUDE)
                .nonKeyAttributes(DESCRIPTION_ATTR, STATUS_ATTR)
                .build();

            final GlobalSecondaryIndex createDateIndex =
GlobalSecondaryIndex.builder()
                .indexName(CREATE_DATE_INDEX)
                .keySchema(createDateKeySchema)
                .projection(createDateProjection)
                .provisionedThroughput(ptIndex)
                .build();

            // TitleIndex
            final List<KeySchemaElement> titleKeySchema = new ArrayList<>();
            titleKeySchema.add(KeySchemaElement.builder()
                .attributeName(TITLE_ATTR)
                .keyType(KeyType.HASH)
                .build());
```

```
            titleKeySchema.add(KeySchemaElement.builder()
                .attributeName(ISSUE_ID_ATTR)
                .keyType(KeyType.RANGE)
                .build());

            final Projection titleProjection =

Projection.builder().projectionType(ProjectionType.KEYS_ONLY).build();

            final GlobalSecondaryIndex titleIndex = GlobalSecondaryIndex.builder()
                .indexName(TITLE_INDEX)
                .keySchema(titleKeySchema)
                .projection(titleProjection)
                .provisionedThroughput(ptIndex)
                .build();

            // DueDateIndex
            final List<KeySchemaElement> dueDateKeySchema = new ArrayList<>();
            dueDateKeySchema.add(KeySchemaElement.builder()
                .attributeName(DUE_DATE_ATTR)
                .keyType(KeyType.HASH)
                .build());

            final Projection dueDateProjection =
                Projection.builder().projectionType(ProjectionType.ALL).build();

            final GlobalSecondaryIndex dueDateIndex = GlobalSecondaryIndex.builder()
                .indexName(DUE_DATE_INDEX)
                .keySchema(dueDateKeySchema)
                .projection(dueDateProjection)
                .provisionedThroughput(ptIndex)
                .build();

            final CreateTableRequest createTableRequest =
CreateTableRequest.builder()
                .tableName(TABLE_NAME)
                .keySchema(tableKeySchema)
                .attributeDefinitions(attributeDefinitions)
                .globalSecondaryIndexes(createDateIndex, titleIndex, dueDateIndex)
                .provisionedThroughput(ProvisionedThroughput.builder()
                    .readCapacityUnits(1L)
                    .writeCapacityUnits(1L)
                    .build())
                .build();
```

```
            System.out.println("Creating table " + TABLE_NAME + "...");
            dynamoDbClient.createTable(createTableRequest);

            // Wait for table to become active
            System.out.println("Waiting for " + TABLE_NAME + " to become
ACTIVE...");
            final DynamoDbWaiter waiter = dynamoDbClient.waiter();
            final DescribeTableRequest describeTableRequest =
                DescribeTableRequest.builder().tableName(TABLE_NAME).build();

            final WaiterResponse<DescribeTableResponse> waiterResponse =
                waiter.waitUntilTableExists(describeTableRequest);
            waiterResponse.matched().response().ifPresent(response ->
System.out.println("Table is now ready for use"));

        } catch (DynamoDbException e) {
            System.err.println("Error creating table: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateTable中的。

创建启用了热吞吐量的表

以下代码示例演示如何创建一个启用了热吞吐量的表。

适用于 Java 的 SDK 2.x

使用 AWS SDK for Java 2.x通过热吞吐量设置创建 DynamoDB 表。

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeDefinition;
import software.amazon.awssdk.services.dynamodb.model.CreateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.CreateTableResponse;
import software.amazon.awssdk.services.dynamodb.model.GlobalSecondaryIndex;
import software.amazon.awssdk.services.dynamodb.model.KeySchemaElement;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.Projection;
import software.amazon.awssdk.services.dynamodb.model.ProvisionedThroughput;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;
```

```
import software.amazon.awssdk.services.dynamodb.model.WarmThroughput;

    public static WarmThroughput buildWarmThroughput(final Long readUnitsPerSecond,
 final Long writeUnitsPerSecond) {
        return WarmThroughput.builder()
            .readUnitsPerSecond(readUnitsPerSecond)
            .writeUnitsPerSecond(writeUnitsPerSecond)
            .build();
    }

    /**
     * Builds a ProvisionedThroughput object with the specified read and write
 capacity units.
     *
     * @param readCapacityUnits The read capacity units
     * @param writeCapacityUnits The write capacity units
     * @return A configured ProvisionedThroughput object
     */
    public static ProvisionedThroughput buildProvisionedThroughput(
        final Long readCapacityUnits, final Long writeCapacityUnits) {
        return ProvisionedThroughput.builder()
            .readCapacityUnits(readCapacityUnits)
            .writeCapacityUnits(writeCapacityUnits)
            .build();
    }

    /**
     * Builds an AttributeDefinition with the specified name and type.
     *
     * @param attributeName The attribute name
     * @param scalarAttributeType The attribute type
     * @return A configured AttributeDefinition
     */
    private static AttributeDefinition buildAttributeDefinition(
        final String attributeName, final ScalarAttributeType scalarAttributeType) {
        return AttributeDefinition.builder()
            .attributeName(attributeName)
            .attributeType(scalarAttributeType)
            .build();
    }

    /**
     * Builds a KeySchemaElement with the specified name and key type.
     *
```

```
     * @param attributeName The attribute name
     * @param keyType The key type (HASH or RANGE)
     * @return A configured KeySchemaElement
     */
    private static KeySchemaElement buildKeySchemaElement(final String
attributeName, final KeyType keyType) {
        return KeySchemaElement.builder()
            .attributeName(attributeName)
            .keyType(keyType)
            .build();
    }

    /**
     * Creates a DynamoDB table with the specified configuration including warm
throughput settings.
     *
     * @param ddb The DynamoDB client
     * @param tableName The name of the table to create
     * @param partitionKey The partition key attribute name
     * @param sortKey The sort key attribute name
     * @param miscellaneousKeyAttribute Additional key attribute name for GSI
     * @param nonKeyAttribute Non-key attribute to include in GSI projection
     * @param tableReadCapacityUnits Read capacity units for the table
     * @param tableWriteCapacityUnits Write capacity units for the table
     * @param tableWarmReadUnitsPerSecond Warm read units per second for the table
     * @param tableWarmWriteUnitsPerSecond Warm write units per second for the table
     * @param globalSecondaryIndexName The name of the GSI to create
     * @param globalSecondaryIndexReadCapacityUnits Read capacity units for the GSI
     * @param globalSecondaryIndexWriteCapacityUnits Write capacity units for the
GSI
     * @param globalSecondaryIndexWarmReadUnitsPerSecond Warm read units per second
for the GSI
     * @param globalSecondaryIndexWarmWriteUnitsPerSecond Warm write units per
second for the GSI
     */
    public static void createDynamoDBTable(
        final DynamoDbClient ddb,
        final String tableName,
        final String partitionKey,
        final String sortKey,
        final String miscellaneousKeyAttribute,
        final String nonKeyAttribute,
        final Long tableReadCapacityUnits,
        final Long tableWriteCapacityUnits,
```

```
        final Long tableWarmReadUnitsPerSecond,
        final Long tableWarmWriteUnitsPerSecond,
        final String globalSecondaryIndexName,
        final Long globalSecondaryIndexReadCapacityUnits,
        final Long globalSecondaryIndexWriteCapacityUnits,
        final Long globalSecondaryIndexWarmReadUnitsPerSecond,
        final Long globalSecondaryIndexWarmWriteUnitsPerSecond) {

        // Define the table attributes
        final AttributeDefinition partitionKeyAttribute =
buildAttributeDefinition(partitionKey, ScalarAttributeType.S);
        final AttributeDefinition sortKeyAttribute =
buildAttributeDefinition(sortKey, ScalarAttributeType.S);
        final AttributeDefinition miscellaneousKeyAttributeDefinition =
            buildAttributeDefinition(miscellaneousKeyAttribute,
ScalarAttributeType.N);
        final AttributeDefinition[] attributeDefinitions = {
            partitionKeyAttribute, sortKeyAttribute,
miscellaneousKeyAttributeDefinition
        };

        // Define the table key schema
        final KeySchemaElement partitionKeyElement =
buildKeySchemaElement(partitionKey, KeyType.HASH);
        final KeySchemaElement sortKeyElement = buildKeySchemaElement(sortKey,
KeyType.RANGE);
        final KeySchemaElement[] keySchema = {partitionKeyElement, sortKeyElement};

        // Define the provisioned throughput for the table
        final ProvisionedThroughput provisionedThroughput =
            buildProvisionedThroughput(tableReadCapacityUnits,
tableWriteCapacityUnits);

        // Define the Global Secondary Index (GSI)
        final KeySchemaElement globalSecondaryIndexPartitionKeyElement =
buildKeySchemaElement(sortKey, KeyType.HASH);
        final KeySchemaElement globalSecondaryIndexSortKeyElement =
            buildKeySchemaElement(miscellaneousKeyAttribute, KeyType.RANGE);
        final KeySchemaElement[] gsiKeySchema = {
            globalSecondaryIndexPartitionKeyElement,
globalSecondaryIndexSortKeyElement
        };

        final Projection gsiProjection = Projection.builder()
```

```
            .projectionType(PROJECTION_TYPE_INCLUDE)
            .nonKeyAttributes(nonKeyAttribute)
            .build();

        final ProvisionedThroughput gsiProvisionedThroughput =
            buildProvisionedThroughput(globalSecondaryIndexReadCapacityUnits,
globalSecondaryIndexWriteCapacityUnits);

        // Define the warm throughput for the Global Secondary Index (GSI)
        final WarmThroughput gsiWarmThroughput = buildWarmThroughput(
            globalSecondaryIndexWarmReadUnitsPerSecond,
globalSecondaryIndexWarmWriteUnitsPerSecond);

        final GlobalSecondaryIndex globalSecondaryIndex =
GlobalSecondaryIndex.builder()
            .indexName(globalSecondaryIndexName)
            .keySchema(gsiKeySchema)
            .projection(gsiProjection)
            .provisionedThroughput(gsiProvisionedThroughput)
            .warmThroughput(gsiWarmThroughput)
            .build();

        // Define the warm throughput for the table
        final WarmThroughput tableWarmThroughput =
            buildWarmThroughput(tableWarmReadUnitsPerSecond,
tableWarmWriteUnitsPerSecond);

        final CreateTableRequest request = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(attributeDefinitions)
            .keySchema(keySchema)
            .provisionedThroughput(provisionedThroughput)
            .globalSecondaryIndexes(globalSecondaryIndex)
            .warmThroughput(tableWarmThroughput)
            .build();

        final CreateTableResponse response = ddb.createTable(request);
        System.out.println(response);
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateTable中的。

创建 Web 应用程序来跟踪 DynamoDB 数据

以下代码示例演示如何创建一个 Web 应用程序，来跟踪 Amazon DynamoDB 表中的工作项，并使用 Amazon Simple Email Service（Amazon SES）来发送报告。

适用于 Java 的 SDK 2.x

展示如何使用 Amazon DynamoDB API 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
- Amazon SES

创建设置了 TTL 的项目

以下代码示例演示如何创建具有 TTL 的项目。

适用于 Java 的 SDK 2.x

```
package com.amazon.samplelib.ttl;

import com.amazon.samplelib.CodeSampleUtils;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.PutItemRequest;
import software.amazon.awssdk.services.dynamodb.model.PutItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.Optional;

/**
 * Creates an item in a DynamoDB table with TTL attributes.
 * This class demonstrates how to add TTL expiration timestamps to DynamoDB items.
 */
```

```
public class CreateTTL {

    private static final String USAGE =
        """
            Usage:
                <tableName> <primaryKey> <sortKey> <region>
            Where:
                tableName - The Amazon DynamoDB table being queried.
                primaryKey - The name of the primary key. Also known as the hash or
 partition key.
                sortKey - The name of the sort key. Also known as the range
 attribute.
                region (optional) - The AWS region that the Amazon DynamoDB table is
 located in. (Default: us-east-1)
            """;
    private static final int DAYS_TO_EXPIRE = 90;
    private static final int SECONDS_PER_DAY = 24 * 60 * 60;
    private static final String PRIMARY_KEY_ATTR = "primaryKey";
    private static final String SORT_KEY_ATTR = "sortKey";
    private static final String CREATION_DATE_ATTR = "creationDate";
    private static final String EXPIRE_AT_ATTR = "expireAt";
    private static final String SUCCESS_MESSAGE = "%s PutItem operation with TTL
 successful.";
    private static final String TABLE_NOT_FOUND_ERROR = "Error: The Amazon DynamoDB
 table \"%s\" can't be found.";

    private final DynamoDbClient dynamoDbClient;

    /**
     * Constructs a CreateTTL instance with the specified DynamoDB client.
     *
     * @param dynamoDbClient The DynamoDB client to use
     */
    public CreateTTL(final DynamoDbClient dynamoDbClient) {
        this.dynamoDbClient = dynamoDbClient;
    }

    /**
     * Constructs a CreateTTL with a default DynamoDB client.
     */
    public CreateTTL() {
        this.dynamoDbClient = null;
    }
```

```
    /**
     * Main method to demonstrate creating an item with TTL.
     *
     * @param args Command line arguments
     */
    public static void main(final String[] args) {
        try {
            int result = new CreateTTL().processArgs(args);
            System.exit(result);
        } catch (Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Process command line arguments and create an item with TTL.
     *
     * @param args Command line arguments
     * @return 0 if successful, non-zero otherwise
     * @throws ResourceNotFoundException If the table doesn't exist
     * @throws DynamoDbException If an error occurs during the operation
     * @throws IllegalArgumentException If arguments are invalid
     */
    public int processArgs(final String[] args) {
        // Argument validation (remove or replace this line when reusing this code)
        CodeSampleUtils.validateArgs(args, new int[] {3, 4}, USAGE);

        final String tableName = args[0];
        final String primaryKey = args[1];
        final String sortKey = args[2];
        final Region region = Optional.ofNullable(args.length > 3 ? args[3] : null)
            .map(Region::of)
            .orElse(Region.US_EAST_1);

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            final CreateTTL createTTL = new CreateTTL(ddb);
            createTTL.createItemWithTTL(tableName, primaryKey, sortKey);
            return 0;
        } catch (Exception e) {
            throw e;
        }
```

```
    }

    /**
     * Creates an item in the specified table with TTL attributes.
     *
     * @param tableName The name of the table
     * @param primaryKeyValue The value for the primary key
     * @param sortKeyValue The value for the sort key
     * @return The response from the PutItem operation
     * @throws ResourceNotFoundException If the table doesn't exist
     * @throws DynamoDbException If an error occurs during the operation
     */
    public PutItemResponse createItemWithTTL(
        final String tableName, final String primaryKeyValue, final String
sortKeyValue) {
        // Get current time in epoch second format
        final long createDate = System.currentTimeMillis() / 1000;

        // Calculate expiration time 90 days from now in epoch second format
        final long expireDate = createDate + (DAYS_TO_EXPIRE * SECONDS_PER_DAY);

        final Map<String, AttributeValue> itemMap = new HashMap<>();
        itemMap.put(
            PRIMARY_KEY_ATTR, AttributeValue.builder().s(primaryKeyValue).build());
        itemMap.put(SORT_KEY_ATTR,
AttributeValue.builder().s(sortKeyValue).build());
        itemMap.put(
            CREATION_DATE_ATTR,
            AttributeValue.builder().n(String.valueOf(createDate)).build());
        itemMap.put(
            EXPIRE_AT_ATTR,
            AttributeValue.builder().n(String.valueOf(expireDate)).build());

        final PutItemRequest request =
            PutItemRequest.builder().tableName(tableName).item(itemMap).build();

        try {
            final PutItemResponse response = dynamoDbClient.putItem(request);
            System.out.println(String.format(SUCCESS_MESSAGE, tableName));
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
```

```
            System.err.println(e.getMessage());
            throw e;
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutItem中的。

创建和管理 MRSC 全局表

以下代码示例显示如何使用多区域强一致性（MRSC）创建和管理 DynamoDB 全局表。

- 创建具有多区域强一致性的表。
- 验证 MRSC 配置和副本状态。
- 通过即时读取来测试各区域间的强一致性。
- 使用 MRSC 保证执行条件写入。
- 清理 MRSC 全局表资源。

适用于 Java 的 SDK 2.x

使用 AWS SDK for Java 2.x创建准备好进行 MRSC 转换的区域表。

```
   public static CreateTableResponse createRegionalTable(final DynamoDbClient
dynamoDbClient, final String tableName) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }

        try {
            LOGGER.info("Creating regional table: " + tableName + " (must be empty
for MRSC)");

            CreateTableRequest createTableRequest = CreateTableRequest.builder()
                .tableName(tableName)
```

```
                .attributeDefinitions(
                    AttributeDefinition.builder()
                        .attributeName("Artist")
                        .attributeType(ScalarAttributeType.S)
                        .build(),
                    AttributeDefinition.builder()
                        .attributeName("SongTitle")
                        .attributeType(ScalarAttributeType.S)
                        .build())
                .keySchema(
                    KeySchemaElement.builder()
                        .attributeName("Artist")
                        .keyType(KeyType.HASH)
                        .build(),
                    KeySchemaElement.builder()
                        .attributeName("SongTitle")
                        .keyType(KeyType.RANGE)
                        .build())
                .billingMode(BillingMode.PAY_PER_REQUEST)
                .build();

            CreateTableResponse response =
dynamoDbClient.createTable(createTableRequest);
            LOGGER.info("Regional table creation initiated. Status: "
                + response.tableDescription().tableStatus());

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to create regional table: " + tableName + " - " +
e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to create regional table: " + tableName)
                .cause(e)
                .build();
        }
    }
```

使用副本将区域表转换为 MRSC 并使用见证。 AWS SDK for Java 2.x

```
    public static UpdateTableResponse convertToMRSCWithWitness(
        final DynamoDbClient dynamoDbClient,
```

```
        final String tableName,
        final Region replicaRegion,
        final Region witnessRegion) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (replicaRegion == null) {
            throw new IllegalArgumentException("Replica region cannot be null");
        }
        if (witnessRegion == null) {
            throw new IllegalArgumentException("Witness region cannot be null");
        }

        try {
            LOGGER.info("Converting table to MRSC with replica in " +
replicaRegion.id() + " and witness in "
                + witnessRegion.id());

            // Create replica update using ReplicationGroupUpdate
            ReplicationGroupUpdate replicaUpdate = ReplicationGroupUpdate.builder()
                .create(CreateReplicationGroupMemberAction.builder()
                    .regionName(replicaRegion.id())
                    .build())
                .build();

            // Create witness update
            GlobalTableWitnessGroupUpdate witnessUpdate =
GlobalTableWitnessGroupUpdate.builder()
                .create(CreateGlobalTableWitnessGroupMemberAction.builder()
                    .regionName(witnessRegion.id())
                    .build())
                .build();

            UpdateTableRequest updateTableRequest = UpdateTableRequest.builder()
                .tableName(tableName)
                .replicaUpdates(List.of(replicaUpdate))
                .globalTableWitnessUpdates(List.of(witnessUpdate))
                .multiRegionConsistency(MultiRegionConsistency.STRONG)
                .build();
```

```
            UpdateTableResponse response =
dynamoDbClient.updateTable(updateTableRequest);
            LOGGER.info("MRSC conversion initiated. Status: "
                + response.tableDescription().tableStatus());
            LOGGER.info("UpdateTableResponse full object: " + response);
            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to convert table to MRSC: " + tableName + " - " +
e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to convert table to MRSC: " + tableName)
                .cause(e)
                .build();
        }
    }
```

使用 AWS SDK for Java 2.x描述 MRSC 全局表配置。

```
    public static DescribeTableResponse describeMRSCTable(final DynamoDbClient
dynamoDbClient, final String tableName) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }

        try {
            LOGGER.info("Describing MRSC global table: " + tableName);

            DescribeTableRequest request =
                DescribeTableRequest.builder().tableName(tableName).build();

            DescribeTableResponse response = dynamoDbClient.describeTable(request);

            LOGGER.info("Table status: " + response.table().tableStatus());
            LOGGER.info("Multi-region consistency: " +
response.table().multiRegionConsistency());
```

```
            if (response.table().replicas() != null
                && !response.table().replicas().isEmpty()) {
                LOGGER.info("Number of replicas: " +
response.table().replicas().size());
                response.table()
                    .replicas()
                    .forEach(replica -> LOGGER.info(
                        "Replica region: " + replica.regionName() + ", Status: " +
replica.replicaStatus()));
            }

            if (response.table().globalTableWitnesses() != null
                && !response.table().globalTableWitnesses().isEmpty()) {
                LOGGER.info("Number of witnesses: "
                    + response.table().globalTableWitnesses().size());
                response.table()
                    .globalTableWitnesses()
                    .forEach(witness -> LOGGER.info(
                        "Witness region: " + witness.regionName() + ", Status: " +
witness.witnessStatus()));
            }

            return response;

        } catch (ResourceNotFoundException e) {
            LOGGER.severe("Table not found: " + tableName + " - " + e.getMessage());
            throw DynamoDbException.builder()
                .message("Table not found: " + tableName)
                .cause(e)
                .build();
        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to describe table: " + tableName + " - " +
e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to describe table: " + tableName)
                .cause(e)
                .build();
        }
    }
```

使用 AWS SDK for Java 2.x添加测试项目以验证 MRSC 强一致性。

```
    public static PutItemResponse putTestItem(
        final DynamoDbClient dynamoDbClient,
        final String tableName,
        final String artist,
        final String songTitle,
        final String album,
        final String year) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (artist == null || artist.trim().isEmpty()) {
            throw new IllegalArgumentException("Artist cannot be null or empty");
        }
        if (songTitle == null || songTitle.trim().isEmpty()) {
            throw new IllegalArgumentException("Song title cannot be null or
empty");
        }

        try {
            LOGGER.info("Adding test item to MRSC global table: " + tableName);

            Map<String, AttributeValue> item = new HashMap<>();
            item.put("Artist", AttributeValue.builder().s(artist).build());
            item.put("SongTitle", AttributeValue.builder().s(songTitle).build());

            if (album != null && !album.trim().isEmpty()) {
                item.put("Album", AttributeValue.builder().s(album).build());
            }
            if (year != null && !year.trim().isEmpty()) {
                item.put("Year", AttributeValue.builder().n(year).build());
            }

            PutItemRequest putItemRequest =
                PutItemRequest.builder().tableName(tableName).item(item).build();

            PutItemResponse response = dynamoDbClient.putItem(putItemRequest);
            LOGGER.info("Test item added successfully with strong consistency");
```

```
            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to add test item to table: " + tableName + " - " +
e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to add test item to table: " + tableName)
                .cause(e)
                .build();
        }
    }
```

使用从 MRSC 副本中读取一致的项目。 AWS SDK for Java 2.x

```
    public static GetItemResponse getItemWithConsistentRead(
        final DynamoDbClient dynamoDbClient, final String tableName, final String
artist, final String songTitle) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (artist == null || artist.trim().isEmpty()) {
            throw new IllegalArgumentException("Artist cannot be null or empty");
        }
        if (songTitle == null || songTitle.trim().isEmpty()) {
            throw new IllegalArgumentException("Song title cannot be null or
empty");
        }

        try {
            LOGGER.info("Reading item from MRSC global table with consistent read: "
+ tableName);

            Map<String, AttributeValue> key = new HashMap<>();
            key.put("Artist", AttributeValue.builder().s(artist).build());
            key.put("SongTitle", AttributeValue.builder().s(songTitle).build());

            GetItemRequest getItemRequest = GetItemRequest.builder()
```

```
                .tableName(tableName)
                .key(key)
                .consistentRead(true)
                .build();

            GetItemResponse response = dynamoDbClient.getItem(getItemRequest);

            if (response.hasItem()) {
                LOGGER.info("Item found with strong consistency - no wait time
needed");
            } else {
                LOGGER.info("Item not found");
            }

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to read item from table: " + tableName + " - " +
e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to read item from table: " + tableName)
                .cause(e)
                .build();
        }
    }
```

使用 AWS SDK for Java 2.x MRSC 保证执行有条件的更新。

```
    public static UpdateItemResponse performConditionalUpdate(
        final DynamoDbClient dynamoDbClient,
        final String tableName,
        final String artist,
        final String songTitle,
        final String rating) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
```

```
        if (artist == null || artist.trim().isEmpty()) {
            throw new IllegalArgumentException("Artist cannot be null or empty");
        }
        if (songTitle == null || songTitle.trim().isEmpty()) {
            throw new IllegalArgumentException("Song title cannot be null or
empty");
        }
        if (rating == null || rating.trim().isEmpty()) {
            throw new IllegalArgumentException("Rating cannot be null or empty");
        }

        try {
            LOGGER.info("Performing conditional update on MRSC global table: " +
tableName);

            Map<String, AttributeValue> key = new HashMap<>();
            key.put("Artist", AttributeValue.builder().s(artist).build());
            key.put("SongTitle", AttributeValue.builder().s(songTitle).build());

            Map<String, String> expressionAttributeNames = new HashMap<>();
            expressionAttributeNames.put("#rating", "Rating");

            Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
            expressionAttributeValues.put(
                ":rating", AttributeValue.builder().n(rating).build());

            UpdateItemRequest updateItemRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #rating = :rating")
                .conditionExpression("attribute_exists(Artist)")
                .expressionAttributeNames(expressionAttributeNames)
                .expressionAttributeValues(expressionAttributeValues)
                .build();

            UpdateItemResponse response =
dynamoDbClient.updateItem(updateItemRequest);
            LOGGER.info("Conditional update successful - demonstrates strong
consistency");

            return response;

        } catch (ConditionalCheckFailedException e) {
            LOGGER.warning("Conditional check failed: " + e.getMessage());
```

```
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to perform conditional update: " + tableName + " -
" + e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to perform conditional update: " + tableName)
                .cause(e)
                .build();
        }
    }
```

等待 MRSC 副本和目击者使用激活。 AWS SDK for Java 2.x

```
    public static void waitForMRSCReplicasActive(
        final DynamoDbClient dynamoDbClient, final String tableName, final int
maxWaitTimeSeconds)
        throws InterruptedException {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (maxWaitTimeSeconds <= 0) {
            throw new IllegalArgumentException("Max wait time must be positive");
        }

        try {
            LOGGER.info("Waiting for MRSC replicas and witnesses to become active: "
+ tableName);

            final long startTime = System.currentTimeMillis();
            final long maxWaitTimeMillis = maxWaitTimeSeconds * 1000L;
            int backoffSeconds = 5; // Start with 5 second intervals
            final int maxBackoffSeconds = 30; // Cap at 30 seconds

            while (System.currentTimeMillis() - startTime < maxWaitTimeMillis) {
                DescribeTableResponse response = describeMRSCTable(dynamoDbClient,
tableName);
```

```
                boolean allActive = true;
                StringBuilder statusReport = new StringBuilder();

                if (response.table().multiRegionConsistency() == null
                    || !MultiRegionConsistency.STRONG
                        .toString()

.equals(response.table().multiRegionConsistency().toString())) {
                    allActive = false;
                    statusReport
                        .append("MultiRegionConsistency: ")
                        .append(response.table().multiRegionConsistency())
                        .append(" ");
                }
                if (response.table().replicas() == null
                    || response.table().replicas().isEmpty()) {
                    allActive = false;
                    statusReport.append("No replicas found. ");
                }
                if (response.table().globalTableWitnesses() == null
                    || response.table().globalTableWitnesses().isEmpty()) {
                    allActive = false;
                    statusReport.append("No witnesses found. ");
                }

                // Check table status
                if (!"ACTIVE".equals(response.table().tableStatus().toString())) {
                    allActive = false;
                    statusReport
                        .append("Table: ")
                        .append(response.table().tableStatus())
                        .append(" ");
                }

                // Check replica status
                if (response.table().replicas() != null) {
                    for (var replica : response.table().replicas()) {
                        if (!"ACTIVE".equals(replica.replicaStatus().toString())) {
                            allActive = false;
                            statusReport
                                .append("Replica(")
                                .append(replica.regionName())
                                .append("): ")
                                .append(replica.replicaStatus())
```

```
                                .append(" ");
                        }
                    }
                }

                // Check witness status
                if (response.table().globalTableWitnesses() != null) {
                    for (var witness : response.table().globalTableWitnesses()) {
                        if (!"ACTIVE".equals(witness.witnessStatus().toString())) {
                            allActive = false;
                            statusReport
                                .append("Witness(")
                                .append(witness.regionName())
                                .append("): ")
                                .append(witness.witnessStatus())
                                .append(" ");
                        }
                    }
                }

                if (allActive) {
                    LOGGER.info("All MRSC replicas and witnesses are now active: " +
tableName);
                    return;
                }

                LOGGER.info("Waiting for MRSC components to become active. Status: "
+ statusReport.toString());
                LOGGER.info("Next check in " + backoffSeconds + " seconds...");

                tempWait(backoffSeconds);

                // Exponential backoff with cap
                backoffSeconds = Math.min(backoffSeconds * 2, maxBackoffSeconds);
            }

            throw DynamoDbException.builder()
                .message("Timeout waiting for MRSC replicas to become active after "
+ maxWaitTimeSeconds + " seconds")
                .build();

        } catch (DynamoDbException | InterruptedException e) {
            LOGGER.severe("Failed to wait for MRSC replicas to become active: " +
tableName + " - " + e.getMessage());
```

```
            throw e;
        }
    }
```

使用清理 MRSC 副本和证人。 AWS SDK for Java 2.x

```
    public static UpdateTableResponse cleanupMRSCReplicas(
        final DynamoDbClient dynamoDbClient,
        final String tableName,
        final Region replicaRegion,
        final Region witnessRegion) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (replicaRegion == null) {
            throw new IllegalArgumentException("Replica region cannot be null");
        }
        if (witnessRegion == null) {
            throw new IllegalArgumentException("Witness region cannot be null");
        }

        try {
            LOGGER.info("Cleaning up MRSC replicas and witnesses for table: " +
tableName);

            // Remove replica using ReplicationGroupUpdate
            ReplicationGroupUpdate replicaUpdate = ReplicationGroupUpdate.builder()
                .delete(DeleteReplicationGroupMemberAction.builder()
                    .regionName(replicaRegion.id())
                    .build())
                .build();

            // Remove witness
            GlobalTableWitnessGroupUpdate witnessUpdate =
GlobalTableWitnessGroupUpdate.builder()
                .delete(DeleteGlobalTableWitnessGroupMemberAction.builder()
                    .regionName(witnessRegion.id())
```

```
                    .build())
                .build();

            UpdateTableRequest updateTableRequest = UpdateTableRequest.builder()
                .tableName(tableName)
                .replicaUpdates(List.of(replicaUpdate))
                .globalTableWitnessUpdates(List.of(witnessUpdate))
                .build();

            UpdateTableResponse response =
dynamoDbClient.updateTable(updateTableRequest);
            LOGGER.info("MRSC cleanup initiated - removing replica and witness.
Response: " + response);

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to cleanup MRSC replicas: " + tableName + " - " +
e.getMessage());
            throw DynamoDbException.builder()
                .message("Failed to cleanup MRSC replicas: " + tableName)
                .cause(e)
                .build();
        }
    }
```

使用 AWS SDK for Java 2.x完成 MRSC 工作流程演示。

```
    public static void demonstrateCompleteMRSCWorkflow(
        final DynamoDbClient primaryClient,
        final DynamoDbClient replicaClient,
        final String tableName,
        final Region replicaRegion,
        final Region witnessRegion)
        throws InterruptedException {

        if (primaryClient == null) {
            throw new IllegalArgumentException("Primary DynamoDB client cannot be
null");
        }
        if (replicaClient == null) {
```

```
            throw new IllegalArgumentException("Replica DynamoDB client cannot be
null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (replicaRegion == null) {
            throw new IllegalArgumentException("Replica region cannot be null");
        }
        if (witnessRegion == null) {
            throw new IllegalArgumentException("Witness region cannot be null");
        }

        try {
            LOGGER.info("=== Starting Complete MRSC Workflow Demonstration ===");

            // Step 1: Create an empty single-Region table
            LOGGER.info("Step 1: Creating empty single-Region table");
            createRegionalTable(primaryClient, tableName);

            // Use the existing GlobalTableOperations method for basic table waiting
            LOGGER.info("Intermediate step: Waiting for table [" + tableName + "] to
become active before continuing");
            GlobalTableOperations.waitForTableActive(primaryClient, tableName);

            // Step 2: Convert to MRSC with replica and witness
            LOGGER.info("Step 2: Converting to MRSC with replica and witness");
            convertToMRSCWithWitness(primaryClient, tableName, replicaRegion,
witnessRegion);

            // Wait for MRSC conversion to complete using MRSC-specific waiter
            LOGGER.info("Waiting for MRSC conversion to complete...");
            waitForMRSCReplicasActive(primaryClient, tableName);

            LOGGER.info("Intermediate step: Waiting for table [" + tableName + "] to
become active before continuing");
            GlobalTableOperations.waitForTableActive(primaryClient, tableName);

            // Step 3: Verify MRSC configuration
            LOGGER.info("Step 3: Verifying MRSC configuration");
            describeMRSCTable(primaryClient, tableName);

            // Step 4: Test strong consistency with data operations
```

```
            LOGGER.info("Step 4: Testing strong consistency with data operations");

            // Add test item to primary region
            putTestItem(primaryClient, tableName, "The Beatles", "Hey Jude", "The
Beatles 1967-1970", "1968");

            // Immediately read from replica region (no wait needed with MRSC)
            LOGGER.info("Reading from replica region immediately (strong
consistency):");
            GetItemResponse getResponse =
                getItemWithConsistentRead(replicaClient, tableName, "The Beatles",
"Hey Jude");

            if (getResponse.hasItem()) {
                LOGGER.info("# Strong consistency verified - item immediately
available in replica region");
            } else {
                LOGGER.warning("# Item not found in replica region");
            }

            // Test conditional update from replica region
            LOGGER.info("Testing conditional update from replica region:");
            performConditionalUpdate(replicaClient, tableName, "The Beatles", "Hey
Jude", "5");
            LOGGER.info("# Conditional update successful - demonstrates strong
consistency");

            // Step 5: Cleanup
            LOGGER.info("Step 5: Cleaning up resources");
            cleanupMRSCReplicas(primaryClient, tableName, replicaRegion,
witnessRegion);

            // Wait for cleanup to complete using basic table waiter
            LOGGER.info("Waiting for replica cleanup to complete...");
            GlobalTableOperations.waitForTableActive(primaryClient, tableName);

            // "Halt" until replica/witness cleanup is complete
            DescribeTableResponse cleanupVerification =
describeMRSCTable(primaryClient, tableName);
            int backoffSeconds = 5; // Start with 5 second intervals
            while (cleanupVerification.table().multiRegionConsistency() != null) {
                LOGGER.info("Waiting additional time (" + backoffSeconds + "
seconds) for MRSC cleanup to complete...");
                tempWait(backoffSeconds);
```

```
                // Exponential backoff with cap
                backoffSeconds = Math.min(backoffSeconds * 2, 30);
                cleanupVerification = describeMRSCTable(primaryClient, tableName);
            }

            // Delete the primary table
            deleteTable(primaryClient, tableName);

            LOGGER.info("=== MRSC Workflow Demonstration Complete ===");
            LOGGER.info("");
            LOGGER.info("Key benefits of Multi-Region Strong Consistency (MRSC):");
            LOGGER.info("- Immediate consistency across all regions (no eventual
consistency delays)");
            LOGGER.info("- Simplified application logic (no need to handle eventual
consistency)");
            LOGGER.info("- Support for conditional writes and transactions across
regions");
            LOGGER.info("- Consistent read operations from any region without
waiting");

        } catch (DynamoDbException | InterruptedException e) {
            LOGGER.severe("MRSC workflow failed: " + e.getMessage());
            throw e;
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - CreateTable
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - UpdateItem
  - UpdateTable

创建和管理用于演示 MREC 的全局表

以下代码示例演示了如何跨多个区域创建和管理 DynamoDB 全局表以及副本。

- 使用全局二级索引和 DynamoDB Streams 创建一个表。
- 在不同区域中添加副本以创建全局表。
- 从全局表中移除副本。
- 添加测试项目以验证跨区域的复制。
- 描述全局表配置和副本状态。

适用于 Java 的 SDK 2.x

使用创建包含全局二级索引和 DynamoDB Streams 的表。 AWS SDK for Java 2.x

```
   public static CreateTableResponse createTableWithGSI(
       final DynamoDbClient dynamoDbClient, final String tableName, final String
indexName) {

       if (dynamoDbClient == null) {
           throw new IllegalArgumentException("DynamoDB client cannot be null");
       }
       if (tableName == null || tableName.trim().isEmpty()) {
           throw new IllegalArgumentException("Table name cannot be null or
empty");
       }
       if (indexName == null || indexName.trim().isEmpty()) {
           throw new IllegalArgumentException("Index name cannot be null or
empty");
       }

       try {
           LOGGER.info("Creating table: " + tableName + " with GSI: " + indexName);

           CreateTableRequest createTableRequest = CreateTableRequest.builder()
               .tableName(tableName)
               .attributeDefinitions(
                   AttributeDefinition.builder()
                       .attributeName("Artist")
                       .attributeType(ScalarAttributeType.S)
                       .build(),
                   AttributeDefinition.builder()
                       .attributeName("SongTitle")
                       .attributeType(ScalarAttributeType.S)
                       .build())
               .keySchema(
```

```
                    KeySchemaElement.builder()
                        .attributeName("Artist")
                        .keyType(KeyType.HASH)
                        .build(),
                    KeySchemaElement.builder()
                        .attributeName("SongTitle")
                        .keyType(KeyType.RANGE)
                        .build())
                .billingMode(BillingMode.PAY_PER_REQUEST)
                .globalSecondaryIndexes(GlobalSecondaryIndex.builder()
                    .indexName(indexName)
                    .keySchema(KeySchemaElement.builder()
                        .attributeName("SongTitle")
                        .keyType(KeyType.HASH)
                        .build())
                    .projection(

Projection.builder().projectionType(ProjectionType.ALL).build())
                    .build())
                .streamSpecification(StreamSpecification.builder()
                    .streamEnabled(true)
                    .streamViewType(StreamViewType.NEW_AND_OLD_IMAGES)
                    .build())
                .build();

            CreateTableResponse response =
dynamoDbClient.createTable(createTableRequest);
            LOGGER.info("Table creation initiated. Status: "
                + response.tableDescription().tableStatus());

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to create table: " + tableName + " - " +
e.getMessage());
            throw e;
        }
    }
```

使用等待表格变为活动状态 AWS SDK for Java 2.x。

```
   public static void waitForTableActive(final DynamoDbClient dynamoDbClient, final
String tableName) {

       if (dynamoDbClient == null) {
           throw new IllegalArgumentException("DynamoDB client cannot be null");
       }
       if (tableName == null || tableName.trim().isEmpty()) {
           throw new IllegalArgumentException("Table name cannot be null or
empty");
       }

       try {
           LOGGER.info("Waiting for table to become active: " + tableName);

           try (DynamoDbWaiter waiter =
               DynamoDbWaiter.builder().client(dynamoDbClient).build()) {
               DescribeTableRequest request =
                   DescribeTableRequest.builder().tableName(tableName).build();

               waiter.waitUntilTableExists(request);
               LOGGER.info("Table is now active: " + tableName);
           }

       } catch (DynamoDbException e) {
           LOGGER.severe("Failed to wait for table to become active: " + tableName
+ " - " + e.getMessage());
           throw e;
       }
   }
```

使用添加副本以创建或扩展全局表 AWS SDK for Java 2.x。

```
   public static UpdateTableResponse addReplica(
       final DynamoDbClient dynamoDbClient,
       final String tableName,
       final Region replicaRegion,
       final String indexName,
       final Long readCapacity) {

       if (dynamoDbClient == null) {
           throw new IllegalArgumentException("DynamoDB client cannot be null");
       }
```

```
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (replicaRegion == null) {
            throw new IllegalArgumentException("Replica region cannot be null");
        }
        if (indexName == null || indexName.trim().isEmpty()) {
            throw new IllegalArgumentException("Index name cannot be null or
empty");
        }
        if (readCapacity == null || readCapacity <= 0) {
            throw new IllegalArgumentException("Read capacity must be a positive
number");
        }

        try {
            LOGGER.info("Adding replica in region: " + replicaRegion.id() + " for
table: " + tableName);

            // Create a ReplicationGroupUpdate for adding a replica
            ReplicationGroupUpdate replicationGroupUpdate =
ReplicationGroupUpdate.builder()
                .create(builder -> builder.regionName(replicaRegion.id())
                    .globalSecondaryIndexes(ReplicaGlobalSecondaryIndex.builder()
                        .indexName(indexName)

.provisionedThroughputOverride(ProvisionedThroughputOverride.builder()
                            .readCapacityUnits(readCapacity)
                            .build())
                        .build())
                    .build())
                .build();

            UpdateTableRequest updateTableRequest = UpdateTableRequest.builder()
                .tableName(tableName)
                .replicaUpdates(replicationGroupUpdate)
                .build();

            UpdateTableResponse response =
dynamoDbClient.updateTable(updateTableRequest);
            LOGGER.info("Replica addition initiated in region: " +
replicaRegion.id());
```

```
            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to add replica in region: " + replicaRegion.id() +
" - " + e.getMessage());
            throw e;
        }
    }
```

使用从全局表中移除副本 AWS SDK for Java 2.x。

```
    public static UpdateTableResponse removeReplica(
        final DynamoDbClient dynamoDbClient, final String tableName, final Region
replicaRegion) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (replicaRegion == null) {
            throw new IllegalArgumentException("Replica region cannot be null");
        }

        try {
            LOGGER.info("Removing replica in region: " + replicaRegion.id() + " for
table: " + tableName);

            // Create a ReplicationGroupUpdate for removing a replica
            ReplicationGroupUpdate replicationGroupUpdate =
ReplicationGroupUpdate.builder()
                .delete(builder -> builder.regionName(replicaRegion.id()).build())
                .build();

            UpdateTableRequest updateTableRequest = UpdateTableRequest.builder()
                .tableName(tableName)
                .replicaUpdates(replicationGroupUpdate)
                .build();
```

```
            UpdateTableResponse response =
dynamoDbClient.updateTable(updateTableRequest);
            LOGGER.info("Replica removal initiated in region: " +
replicaRegion.id());

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to remove replica in region: " +
replicaRegion.id() + " - " + e.getMessage());
            throw e;
        }
    }
```

使用添加测试项目以验证复制 AWS SDK for Java 2.x。

```
    public static PutItemResponse putTestItem(
        final DynamoDbClient dynamoDbClient, final String tableName, final String
artist, final String songTitle) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }
        if (artist == null || artist.trim().isEmpty()) {
            throw new IllegalArgumentException("Artist cannot be null or empty");
        }
        if (songTitle == null || songTitle.trim().isEmpty()) {
            throw new IllegalArgumentException("Song title cannot be null or
empty");
        }

        try {
            LOGGER.info("Adding test item to table: " + tableName);

            Map<String,
software.amazon.awssdk.services.dynamodb.model.AttributeValue> item = new
HashMap<>();
            item.put(
```

```
                "Artist",

software.amazon.awssdk.services.dynamodb.model.AttributeValue.builder()
                    .s(artist)
                    .build());
            item.put(
                "SongTitle",

software.amazon.awssdk.services.dynamodb.model.AttributeValue.builder()
                    .s(songTitle)
                    .build());

            PutItemRequest putItemRequest =
                PutItemRequest.builder().tableName(tableName).item(item).build();

            PutItemResponse response = dynamoDbClient.putItem(putItemRequest);
            LOGGER.info("Test item added successfully");

            return response;

        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to add test item to table: " + tableName + " - " +
e.getMessage());
            throw e;
        }
    }
```

使用 AWS SDK for Java 2.x描述全局表配置和副本。

```
    public static DescribeTableResponse describeTable(final DynamoDbClient
dynamoDbClient, final String tableName) {

        if (dynamoDbClient == null) {
            throw new IllegalArgumentException("DynamoDB client cannot be null");
        }
        if (tableName == null || tableName.trim().isEmpty()) {
            throw new IllegalArgumentException("Table name cannot be null or
empty");
        }

        try {
            LOGGER.info("Describing table: " + tableName);
```

```
            DescribeTableRequest request =
                DescribeTableRequest.builder().tableName(tableName).build();

            DescribeTableResponse response = dynamoDbClient.describeTable(request);

            LOGGER.info("Table status: " + response.table().tableStatus());
            if (response.table().replicas() != null
                && !response.table().replicas().isEmpty()) {
                LOGGER.info("Number of replicas: " +
response.table().replicas().size());
                response.table()
                    .replicas()
                    .forEach(replica -> LOGGER.info(
                        "Replica region: " + replica.regionName() + ", Status: " +
replica.replicaStatus()));
            }

            return response;

        } catch (ResourceNotFoundException e) {
            LOGGER.severe("Table not found: " + tableName + " - " + e.getMessage());
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.severe("Failed to describe table: " + tableName + " - " +
e.getMessage());
            throw e;
        }
    }
```

使用进行全局表操作的完整示例 AWS SDK for Java 2.x。

```
    public static void exampleUsage(final Region sourceRegion, final Region
replicaRegion) {

        String tableName = "Music";
        String indexName = "SongTitleIndex";
        Long readCapacity = 15L;

        // Create DynamoDB client for the source region
        try (DynamoDbClient dynamoDbClient =
            DynamoDbClient.builder().region(sourceRegion).build()) {
```

```
            try {
                // Step 1: Create the initial table with GSI and streams
                LOGGER.info("Step 1: Creating table in source region: " +
sourceRegion.id());
                createTableWithGSI(dynamoDbClient, tableName, indexName);

                // Step 2: Wait for table to become active
                LOGGER.info("Step 2: Waiting for table to become active");
                waitForTableActive(dynamoDbClient, tableName);

                // Step 3: Add replica in destination region
                LOGGER.info("Step 3: Adding replica in region: " +
replicaRegion.id());
                addReplica(dynamoDbClient, tableName, replicaRegion, indexName,
readCapacity);

                // Step 4: Wait a moment for replica creation to start
                Thread.sleep(5000);

                // Step 5: Describe table to view replica information
                LOGGER.info("Step 5: Describing table to view replicas");
                describeTable(dynamoDbClient, tableName);

                // Step 6: Add a test item to verify replication
                LOGGER.info("Step 6: Adding test item to verify replication");
                putTestItem(dynamoDbClient, tableName, "TestArtist", "TestSong");

                LOGGER.info("Global table setup completed successfully!");
                LOGGER.info("You can verify replication by checking the item in
region: " + replicaRegion.id());

                // Step 7: Remove replica and clean up table
                LOGGER.info("Step 7: Removing replica from region: " +
replicaRegion.id());
                removeReplica(dynamoDbClient, tableName, replicaRegion);
                DeleteTableResponse deleteTableResponse =
dynamoDbClient.deleteTable(
                    DeleteTableRequest.builder().tableName(tableName).build());
                LOGGER.info("MREC global table demonstration completed
successfully!");

            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
```

```
                throw new RuntimeException("Thread was interrupted", e);
            } catch (DynamoDbException e) {
                LOGGER.severe("DynamoDB operation failed: " + e.getMessage());
                throw e;
            }
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - CreateTable
  - DescribeTable
  - PutItem
  - UpdateTable

检测图像中的 PPE

以下代码示例展示如何构建采用 Amazon Rekognition 来检测图像中的个人防护设备（PPE）的应用程序。

适用于 Java 的 SDK 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
- Amazon SES

监控 DynamoDB 性能

以下代码示例显示如何配置应用程序使用 DynamoDB 来监控性能。

适用于 Java 的 SDK 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

执行高级查询操作

以下代码示例演示如何在 DynamoDB 中执行高级查询操作。

- 使用各种筛选和条件技术查询表。
- 对大型结果集实现分页。
- 使用全局二级索引实现备选访问模式。
- 根据应用程序要求应用一致性控制。

适用于 Java 的 SDK 2.x

使用强一致性读取进行查询 AWS SDK for Java 2.x。

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithConsistentReads(
        final String tableName,
        final String partitionKeyName,
```

```
        final String partitionKeyValue,
        final boolean useConsistentRead) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .consistentRead(useConsistentRead)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query successful. Found {0} items",
response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with consistent reads", e);
            throw e;
        }
    }
```

使用带的全局二级索引进行查询 AWS SDK for Java 2.x。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryTable(
        final String tableName, final String partitionKeyName, final String
 partitionKeyValue) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query on base table successful. Found " +
 response.count() + " items");
            return response;
```

```
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw new DynamoDbQueryException("Table not found: " + tableName, e);
        } catch (DynamoDbException e) {
            System.err.println("Error querying base table: " + e.getMessage());
            throw new DynamoDbQueryException("Failed to execute query on base
table", e);
        }
    }

    /**
     * Queries a DynamoDB Global Secondary Index (GSI) by partition key.
     *
     * @param tableName         The name of the DynamoDB table
     * @param indexName         The name of the GSI
     * @param partitionKeyName  The name of the GSI partition key attribute
     * @param partitionKeyValue The value of the GSI partition key to query
     * @return The query response from DynamoDB
     * @throws ResourceNotFoundException if the table or index doesn't exist
     * @throws DynamoDbException if the query fails
     */
    public QueryResponse queryGlobalSecondaryIndex(
        final String tableName, final String indexName, final String
partitionKeyName, final String partitionKeyValue) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Index name", indexName);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_IK,
partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_IK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
```

```
            .tableName(tableName)
            .indexName(indexName)
            .keyConditionExpression(GSI_KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query on GSI successful. Found " + response.count()
 + " items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format(
                "Error: The Amazon DynamoDB table \"%s\" or index \"%s\" can't be
 found.\n", tableName, indexName);
            throw new DynamoDbQueryException("Table or index not found: " +
 tableName + "/" + indexName, e);
        } catch (DynamoDbException e) {
            System.err.println("Error querying GSI: " + e.getMessage());
            throw new DynamoDbQueryException("Failed to execute query on GSI", e);
        }
    }
```

使用 AWS SDK for Java 2.x分页查询。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    public List<Map<String, AttributeValue>> queryWithPagination(
        final String tableName, final String partitionKeyName, final String
 partitionKeyValue, final int pageSize) {
```

```
        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validatePositiveInteger("Page size", pageSize);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        QueryRequest.Builder queryRequestBuilder = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .limit(pageSize);

        // List to store all items from all pages
        final List<Map<String, AttributeValue>> allItems = new ArrayList<>();

        // Map to store the last evaluated key for pagination
        Map<String, AttributeValue> lastEvaluatedKey = null;
        int pageNumber = 1;

        try {
            do {
                // If we have a last evaluated key, use it for the next page
                if (lastEvaluatedKey != null) {
                    queryRequestBuilder.exclusiveStartKey(lastEvaluatedKey);
                }

                // Execute the query
                final QueryResponse response =
dynamoDbClient.query(queryRequestBuilder.build());

                // Process the current page of results
```

```
                final List<Map<String, AttributeValue>> pageItems =
response.items();
                allItems.addAll(pageItems);

                // Get the last evaluated key for the next page
                lastEvaluatedKey = response.lastEvaluatedKey();
                if (lastEvaluatedKey != null && lastEvaluatedKey.isEmpty()) {
                    lastEvaluatedKey = null;
                }

                System.out.println("Page " + pageNumber + ": Retrieved " +
pageItems.size() + " items (Running total: "
                    + allItems.size() + ")");

                pageNumber++;

            } while (lastEvaluatedKey != null);

            System.out.println("Query with pagination complete. Retrieved a total of
" + allItems.size()
                + " items across " + (pageNumber - 1) + " pages");

            return allItems;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with pagination: " + e.getMessage());
            throw e;
        }
    }
```

使用复杂的过滤器进行查询 AWS SDK for Java 2.x。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
```

```
import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithComplexFilter(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String statusAttrName,
        final String activeStatus,
        final String pendingStatus,
        final String priceAttrName,
        final double minPrice,
        final double maxPrice,
        final String categoryAttrName) {

        // Validate parameters
        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Status attribute name",
 statusAttrName);
        CodeSampleUtils.validateStringParameter("Active status", activeStatus);
        CodeSampleUtils.validateStringParameter("Pending status", pendingStatus);
        CodeSampleUtils.validateStringParameter("Price attribute name",
 priceAttrName);
        CodeSampleUtils.validateStringParameter("Category attribute name",
 categoryAttrName);
        CodeSampleUtils.validateNumericRange("Minimum price", minPrice, 0.0,
 Double.MAX_VALUE);
        CodeSampleUtils.validateNumericRange("Maximum price", maxPrice, minPrice,
 Double.MAX_VALUE);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#pk", partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_STATUS,
 statusAttrName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PRICE,
 priceAttrName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_CATEGORY,
 categoryAttrName);
```

```
        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            ":pkValue", AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_ACTIVE,
            AttributeValue.builder().s(activeStatus).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PENDING,
            AttributeValue.builder().s(pendingStatus).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_MIN_PRICE,
            AttributeValue.builder().n(String.valueOf(minPrice)).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_MAX_PRICE,
            AttributeValue.builder().n(String.valueOf(maxPrice)).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(FILTER_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        return dynamoDbClient.query(queryRequest);
    }
```

使用动态构造的过滤器表达式进行查询 AWS SDK for Java 2.x。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
```

```
   public static QueryResponse queryWithDynamicFilter(
       final String tableName,
       final String partitionKeyName,
       final String partitionKeyValue,
       final Map<String, Object> filterCriteria,
       final Region region,
       final DynamoDbClient dynamoDbClient) {

       validateParameters(tableName, partitionKeyName, partitionKeyValue,
filterCriteria);

       DynamoDbClient ddbClient = dynamoDbClient;
       boolean shouldClose = false;

       try {
           if (ddbClient == null) {
               ddbClient = createClient(region);
               shouldClose = true;
           }

           final QueryWithDynamicFilter queryHelper = new
QueryWithDynamicFilter(ddbClient);
           return queryHelper.queryWithDynamicFilter(tableName, partitionKeyName,
partitionKeyValue, filterCriteria);
       } catch (ResourceNotFoundException e) {
           System.err.println("Table not found: " + tableName);
           throw e;
       } catch (DynamoDbException e) {
           System.err.println("Failed to execute dynamic filter query: " +
e.getMessage());
           throw e;
       } catch (Exception e) {
           System.err.println("Unexpected error during query: " + e.getMessage());
           throw e;
       } finally {
           if (shouldClose && ddbClient != null) {
               ddbClient.close();
           }
       }
   }

   public static void main(String[] args) {
       final String usage =
```

```
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue>
<filterAttrName> <filterAttrValue> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    filterAttrName - The name of the attribute to filter on.
                    filterAttrValue - The value to filter by.
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 5) {
            System.out.println(usage);
            System.exit(1);
        }

        final String tableName = args[0];
        final String partitionKeyName = args[1];
        final String partitionKeyValue = args[2];
        final String filterAttrName = args[3];
        final String filterAttrValue = args[4];
        final Region region = args.length > 5 ? Region.of(args[5]) :
Region.US_EAST_1;

        System.out.println("Querying items with dynamic filter: " + filterAttrName +
" = " + filterAttrValue);

        try {
            // Using the builder pattern to create and execute the query
            final QueryResponse response = new DynamicFilterQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withFilterCriterion(filterAttrName, filterAttrValue)
                .withRegion(region)
                .execute();

            // Process the results
            System.out.println("Found " + response.count() + " items:");
            response.items().forEach(item -> System.out.println(item));
```

```
            // Demonstrate multiple filter criteria
            System.out.println("\nNow querying with multiple filter criteria:");

            Map<String, Object> multipleFilters = new HashMap<>();
            multipleFilters.put(filterAttrName, filterAttrValue);
            multipleFilters.put("status", "active");

            final QueryResponse multiFilterResponse = new
 DynamicFilterQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withFilterCriteria(multipleFilters)
                .withRegion(region)
                .execute();

            System.out.println("Found " + multiFilterResponse.count() + " items with
 multiple filters:");
            multiFilterResponse.items().forEach(item -> System.out.println(item));

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

使用筛选表达式进行查询，并使用限制 AWS SDK for Java 2.x。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
```

```
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithFilterAndLimit(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String filterAttrName,
        final String filterAttrValue,
        final int limit) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Filter attribute name",
 filterAttrName);
        CodeSampleUtils.validateStringParameter("Filter attribute value",
 filterAttrValue);
        CodeSampleUtils.validatePositiveInteger("Limit", limit);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_FILTER,
 filterAttrName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_FILTER,
            AttributeValue.builder().s(filterAttrValue).build());

        // Create the filter expression
        final String filterExpression = "#filterAttr = :filterValue";
```

```
        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(filterExpression)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .limit(limit)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query with filter and limit successful. Found
{0} items", response.count());
            LOGGER.log(
                Level.INFO, "ScannedCount: {0} (total items evaluated before
filtering)", response.scannedCount());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with filter and limit: {0}",
e.getMessage());
            throw e;
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 Query。

执行列表操作

以下代码示例演示如何在 DynamoDB 中执行列表操作。

- 向列表属性添加元素。
- 从列表属性中移除元素。
- 按索引更新列表中的特定元素。
- 使用列表追加和列表索引函数。

适用于 Java 的 SDK 2.x

使用演示列表操作 AWS SDK for Java 2.x。

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    /**
     * Appends items to a list attribute.
     *
     * <p>This method demonstrates how to use the list_append function to add
     * items to the end of a list attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param listAttributeName The name of the list attribute
     * @param itemsToAppend The items to append to the list
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse appendToList(
            DynamoDbClient dynamoDbClient,
            String tableName,
            Map<String, AttributeValue> key,
            String listAttributeName,
            List<AttributeValue> itemsToAppend) {

        // Create a list value from the items to append
        AttributeValue listValue =
 AttributeValue.builder().l(itemsToAppend).build();

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
```

```
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attrName =
list_append(if_not_exists(#attrName, :emptyList), :newItems)")
            .expressionAttributeNames(Map.of("#attrName", listAttributeName))
            .expressionAttributeValues(Map.of(
                ":newItems",
                listValue,
                ":emptyList",
                AttributeValue.builder().l(new
ArrayList<AttributeValue>()).build()))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Prepends items to a list attribute.
     *
     * <p>This method demonstrates how to use the list_append function to add
     * items to the beginning of a list attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param listAttributeName The name of the list attribute
     * @param itemsToPrepend The items to prepend to the list
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse prependToList(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String listAttributeName,
        List<AttributeValue> itemsToPrepend) {

        // Create a list value from the items to prepend
        AttributeValue listValue =
AttributeValue.builder().l(itemsToPrepend).build();

        // Define the update parameters
```

```
        // Note: To prepend, we put the new items first in the list_append function
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attrName = list_append(:newItems,
if_not_exists(#attrName, :emptyList))")
            .expressionAttributeNames(Map.of("#attrName", listAttributeName))
            .expressionAttributeValues(Map.of(
                ":newItems",
                listValue,
                ":emptyList",
                AttributeValue.builder().l(new
ArrayList<AttributeValue>()).build()))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Updates a specific element in a list attribute.
     *
     * <p>This method demonstrates how to update a specific element in a list
     * by its index.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param listAttributeName The name of the list attribute
     * @param index The index of the element to update
     * @param newValue The new value for the element
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateListElement(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String listAttributeName,
        int index,
        AttributeValue newValue) {

        // Define the update parameters
```

```
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attrName[" + index + "] = :newValue")
            .expressionAttributeNames(Map.of("#attrName", listAttributeName))
            .expressionAttributeValues(Map.of(":newValue", newValue))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Removes a specific element from a list attribute.
     *
     * <p>This method demonstrates how to remove a specific element from a list
     * by its index.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param listAttributeName The name of the list attribute
     * @param index The index of the element to remove
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse removeListElement(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String listAttributeName,
        int index) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("REMOVE #attrName[" + index + "]")
            .expressionAttributeNames(Map.of("#attrName", listAttributeName))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
```

```
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Gets the current value of a list attribute.
     *
     * <p>Helper method to retrieve the current value of a list attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @param listAttributeName The name of the list attribute
     * @return The list attribute value or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static List<AttributeValue> getListAttribute(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, String listAttributeName) {

        // Define the get parameters
        GetItemRequest request = GetItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .projectionExpression(listAttributeName)
            .build();

        try {
            // Perform the get operation
            GetItemResponse response = dynamoDbClient.getItem(request);

            // Return the list attribute if it exists, otherwise null
            if (response.item() != null &&
response.item().containsKey(listAttributeName)) {
                return response.item().get(listAttributeName).l();
            }

            return null;
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to get list attribute: " + e.getMessage())
                .cause(e)
                .build();
        }
    }
```

列表操作的用法示例 AWS SDK for Java 2.x。

```
   public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
       // Example key
       Map<String, AttributeValue> key = new HashMap<>();
       key.put("ProductId", AttributeValue.builder().s("P12345").build());

       System.out.println("Demonstrating list operations in DynamoDB");

       try {
           // Example 1: Append items to a list
           System.out.println("\nExample 1: Appending items to a list");
           List<AttributeValue> tagsToAppend = List.of(
               AttributeValue.builder().s("Electronics").build(),
               AttributeValue.builder().s("Gadget").build());

           UpdateItemResponse appendResponse = appendToList(dynamoDbClient,
tableName, key, "Tags", tagsToAppend);

           System.out.println("Updated list attribute: " +
appendResponse.attributes());

           // Example 2: Prepend items to a list
           System.out.println("\nExample 2: Prepending items to a list");
           List<AttributeValue> tagsToPrepend = List.of(
               AttributeValue.builder().s("Featured").build(),
               AttributeValue.builder().s("New").build());

           UpdateItemResponse prependResponse = prependToList(dynamoDbClient,
tableName, key, "Tags", tagsToPrepend);

           System.out.println("Updated list attribute: " +
prependResponse.attributes());

           // Example 3: Update a specific element in a list
           System.out.println("\nExample 3: Updating a specific element in a
list");
           UpdateItemResponse updateResponse = updateListElement(
               dynamoDbClient,
               tableName,
```

```
                key,
                "Tags",
                0,
                AttributeValue.builder().s("BestSeller").build());

            System.out.println("Updated list attribute: " +
updateResponse.attributes());

            // Example 4: Remove a specific element from a list
            System.out.println("\nExample 4: Removing a specific element from a
list");
            UpdateItemResponse removeResponse = removeListElement(dynamoDbClient,
tableName, key, "Tags", 1);

            System.out.println("Updated list attribute: " +
removeResponse.attributes());

            // Example 5: Get the current value of a list attribute
            System.out.println("\nExample 5: Getting the current value of a list
attribute");
            List<AttributeValue> currentList = getListAttribute(dynamoDbClient,
tableName, key, "Tags");

            if (currentList != null) {
                System.out.println("Current list attribute:");
                for (int i = 0; i < currentList.size(); i++) {
                    System.out.println("  [" + i + "]: " + currentList.get(i).s());
                }
            } else {
                System.out.println("List attribute not found");
            }

            // Explain list operations
            System.out.println("\nKey points about DynamoDB list operations:");
            System.out.println("1. Lists are ordered collections of attributes");
            System.out.println("2. Use list_append to add items to a list");
            System.out.println("3. To append items, use list_append(existingList,
newItems)");
            System.out.println("4. To prepend items, use list_append(newItems,
existingList)");
            System.out.println("5. Use index notation (list[0]) to access or update
specific elements");
            System.out.println("6. Use REMOVE to delete elements from a list");
            System.out.println("7. List indices are zero-based");
```

```
            System.out.println("8. Use if_not_exists to handle the case where the
 list doesn't exist yet");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考UpdateItem中的。

执行映射操作

以下代码示例演示如何在 DynamoDB 中执行映射操作。

- 在映射结构中添加和更新嵌套属性。
- 从映射中移除特定字段。
- 使用深度嵌套的映射属性。

适用于 Java 的 SDK 2.x

使用演示地图操作 AWS SDK for Java 2.x。

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

    /**
     * Updates a map attribute that may not exist.
     *
     * <p>This method demonstrates how to safely update a map attribute
     * by using if_not_exists to handle the case where the map doesn't exist yet.
```

```
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param mapName The name of the map attribute
     * @param mapKey The key within the map to update
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateMapAttributeSafe(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String mapName,
        String mapKey,
        AttributeValue value) {

        // Create an empty map to use if the map doesn't exist
        Map<String, AttributeValue> emptyMap = new HashMap<>();
        AttributeValue emptyMapValue = AttributeValue.builder().m(emptyMap).build();

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #mapName = if_not_exists(#mapName, :emptyMap),
#mapName.#mapKey = :value")
            .expressionAttributeNames(Map.of(
                "#mapName", mapName,
                "#mapKey", mapKey))
            .expressionAttributeValues(Map.of(
                ":value",
                value,
                ":emptyMap",
                AttributeValue.builder().m(new HashMap<>()).build()))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
```

```
     * Adds an attribute to a nested map.
     *
     * <p>This method demonstrates how to update a nested attribute without
     * overwriting the entire map.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param path The path to the nested attribute as a list
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse addToNestedMap(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        List<String> path,
        AttributeValue value) {

        // Create expression attribute names for each part of the path
        Map<String, String> expressionAttributeNames = new HashMap<>();
        for (int i = 0; i < path.size(); i++) {
            expressionAttributeNames.put("#attr" + i, path.get(i));
        }

        // Build the attribute path using the expression attribute names
        StringBuilder attributePathExpression = new StringBuilder();
        for (int i = 0; i < path.size(); i++) {
            if (i > 0) {
                attributePathExpression.append(".");
            }
            attributePathExpression.append("#attr").append(i);
        }

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET " + attributePathExpression.toString() + "
= :value")
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
```

```
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Removes an attribute from a map.
     *
     * <p>This method demonstrates how to remove a specific attribute from a map.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param mapName The name of the map attribute
     * @param mapKey The key within the map to remove
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse removeMapAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String mapName,
        String mapKey) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("REMOVE #mapName.#mapKey")
            .expressionAttributeNames(Map.of(
                "#mapName", mapName,
                "#mapKey", mapKey))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Creates a map with multiple attributes in a single operation.
     *
```

```
     * <p>This method demonstrates how to create a map with multiple attributes
     * in a single update operation.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param mapName The name of the map attribute
     * @param attributes The attributes to set in the map
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse createMapWithAttributes(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String mapName,
        Map<String, AttributeValue> attributes) {

        // Create a map value from the attributes
        AttributeValue mapValue = AttributeValue.builder().m(attributes).build();

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #mapName = :mapValue")
            .expressionAttributeNames(Map.of("#mapName", mapName))
            .expressionAttributeValues(Map.of(":mapValue", mapValue))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Gets the current value of a map attribute.
     *
     * <p>Helper method to retrieve the current value of a map attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @param mapName The name of the map attribute
```

```
     * @return The map attribute value or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Map<String, AttributeValue> getMapAttribute(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, String mapName) {

        // Define the get parameters
        GetItemRequest request = GetItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .projectionExpression(mapName)
            .build();

        try {
            // Perform the get operation
            GetItemResponse response = dynamoDbClient.getItem(request);

            // Return the map attribute if it exists, otherwise null
            if (response.item() != null && response.item().containsKey(mapName)) {
                return response.item().get(mapName).m();
            }

            return null;
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to get map attribute: " + e.getMessage())
                .cause(e)
                .build();
        }
    }
```

地图操作的用法示例 AWS SDK for Java 2.x。

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating map operations in DynamoDB");
```

```
        try {
            // Example 1: Create a map with multiple attributes
            System.out.println("\nExample 1: Creating a map with multiple
attributes");
            Map<String, AttributeValue> productDetails = new HashMap<>();
            productDetails.put("Color", AttributeValue.builder().s("Red").build());
            productDetails.put("Weight", AttributeValue.builder().n("2.5").build());
            productDetails.put(
                "Dimensions", AttributeValue.builder().s("10x20x5").build());

            UpdateItemResponse createResponse =
                createMapWithAttributes(dynamoDbClient, tableName, key, "Details",
productDetails);

            System.out.println("Created map attribute: " +
createResponse.attributes());

            // Example 2: Update a specific attribute in a map
            System.out.println("\nExample 2: Updating a specific attribute in a
map");
            UpdateItemResponse updateResponse = updateMapAttributeSafe(
                dynamoDbClient,
                tableName,
                key,
                "Details",
                "Color",
                AttributeValue.builder().s("Blue").build());

            System.out.println("Updated map attribute: " +
updateResponse.attributes());

            // Example 3: Add an attribute to a nested map
            System.out.println("\nExample 3: Adding an attribute to a nested map");
            UpdateItemResponse nestedResponse = addToNestedMap(
                dynamoDbClient,
                tableName,
                key,
                List.of("Specifications", "Technical", "Resolution"),
                AttributeValue.builder().s("1920x1080").build());

            System.out.println("Added to nested map: " +
nestedResponse.attributes());

            // Example 4: Remove an attribute from a map
```

```
            System.out.println("\nExample 4: Removing an attribute from a map");
            UpdateItemResponse removeResponse =
                removeMapAttribute(dynamoDbClient, tableName, key, "Details",
"Dimensions");

            System.out.println("Updated map after removal: " +
removeResponse.attributes());

            // Example 5: Get the current value of a map attribute
            System.out.println("\nExample 5: Getting the current value of a map
attribute");
            Map<String, AttributeValue> currentMap = getMapAttribute(dynamoDbClient,
tableName, key, "Details");

            if (currentMap != null) {
                System.out.println("Current map attribute:");
                for (Map.Entry<String, AttributeValue> entry :
currentMap.entrySet()) {
                    System.out.println("  " + entry.getKey() + ": " +
entry.getValue());
                }
            } else {
                System.out.println("Map attribute not found");
            }

            // Explain map operations
            System.out.println("\nKey points about DynamoDB map operations:");
            System.out.println("1. Maps are unordered collections of name-value
pairs");
            System.out.println("2. Use dot notation (map.key) to access or update
specific attributes");
            System.out.println("3. You can update individual attributes without
overwriting the entire map");
            System.out.println("4. Maps can be nested to create complex data
structures");
            System.out.println("5. Use REMOVE to delete attributes from a map");
            System.out.println("6. You can create a map with multiple attributes in
a single operation");
            System.out.println("7. Map keys are case-sensitive");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
```

```
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考UpdateItem中的。

执行设置操作

以下代码示例演示如何在 DynamoDB 中执行设置操作。

- 向设置属性添加元素。
- 从设置属性中移除元素。
- 将 ADD 和 DELETE 操作用于设置。

适用于 Java 的 SDK 2.x

使用演示集合运算 AWS SDK for Java 2.x。

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ReturnValue;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;

    /**
     * Adds values to a string set attribute.
     *
     * <p>This method demonstrates how to use the ADD operation to add values
     * to a string set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param setAttributeName The name of the set attribute
```

```
     * @param valuesToAdd The values to add to the set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse addToStringSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        Set<String> valuesToAdd) {

        // Create a string set value from the values to add
        AttributeValue setValue = AttributeValue.builder().ss(valuesToAdd).build();

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("ADD #setAttr :valuesToAdd")
            .expressionAttributeNames(Map.of("#setAttr", setAttributeName))
            .expressionAttributeValues(Map.of(":valuesToAdd", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Adds values to a number set attribute.
     *
     * <p>This method demonstrates how to use the ADD operation to add values
     * to a number set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param setAttributeName The name of the set attribute
     * @param valuesToAdd The values to add to the set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse addToNumberSet(
        DynamoDbClient dynamoDbClient,
```

```
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        Set<Number> valuesToAdd) {

        // Convert numbers to strings for DynamoDB
        Set<String> stringValues = new HashSet<>();
        for (Number value : valuesToAdd) {
            stringValues.add(value.toString());
        }

        // Create a number set value from the values to add
        AttributeValue setValue = AttributeValue.builder().ns(stringValues).build();

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("ADD #setAttr :valuesToAdd")
            .expressionAttributeNames(Map.of("#setAttr", setAttributeName))
            .expressionAttributeValues(Map.of(":valuesToAdd", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Removes values from a set attribute.
     *
     * <p>This method demonstrates how to use the DELETE operation to remove values
     * from a set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param setAttributeName The name of the set attribute
     * @param valuesToRemove The values to remove from the set
     * @param isNumberSet Whether the set is a number set (true) or string set
(false)
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
```

```
    public static UpdateItemResponse removeFromSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        Set<?> valuesToRemove,
        boolean isNumberSet) {

        AttributeValue setValue;

        if (isNumberSet) {
            // Convert numbers to strings for DynamoDB
            Set<String> stringValues = new HashSet<>();
            for (Object value : valuesToRemove) {
                if (value instanceof Number) {
                    stringValues.add(value.toString());
                } else {
                    throw new IllegalArgumentException("Values must be numbers for a
number set");
                }
            }

            setValue = AttributeValue.builder().ns(stringValues).build();
        } else {
            // Convert objects to strings for DynamoDB
            Set<String> stringValues = new HashSet<>();
            for (Object value : valuesToRemove) {
                stringValues.add(value.toString());
            }

            setValue = AttributeValue.builder().ss(stringValues).build();
        }

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("DELETE #setAttr :valuesToRemove")
            .expressionAttributeNames(Map.of("#setAttr", setAttributeName))
            .expressionAttributeValues(Map.of(":valuesToRemove", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
```

```
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Checks if a value exists in a set attribute.
     *
     * <p>This method demonstrates how to use the contains function to check
     * if a value exists in a set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to check
     * @param setAttributeName The name of the set attribute
     * @param valueToCheck The value to check for
     * @return Map containing the result of the check
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Map<String, Object> checkIfValueInSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        String valueToCheck) {

        Map<String, Object> result = new HashMap<>();

        try {
            // Define the update parameters with a condition expression
            UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #tempAttr = :tempVal")
                .conditionExpression("contains(#setAttr, :valueToCheck)")
                .expressionAttributeNames(Map.of("#setAttr", setAttributeName,
"#tempAttr", "TempAttribute"))
                .expressionAttributeValues(Map.of(
                    ":valueToCheck",
AttributeValue.builder().s(valueToCheck).build(),
                    ":tempVal", AttributeValue.builder().s("TempValue").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            // Attempt the update operation
            dynamoDbClient.updateItem(request);
```

```
            // If we get here, the condition was met
            result.put("exists", true);
            result.put("message", "Value '" + valueToCheck + "' exists in the set");

            // Clean up the temporary attribute
            UpdateItemRequest cleanupRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("REMOVE #tempAttr")
                .expressionAttributeNames(Map.of("#tempAttr", "TempAttribute"))
                .build();

            dynamoDbClient.updateItem(cleanupRequest);

        } catch (DynamoDbException e) {
            if (e.getMessage().contains("ConditionalCheckFailed")) {
                // The condition was not met
                result.put("exists", false);
                result.put("message", "Value '" + valueToCheck + "' does not exist
in the set");
            } else {
                // Some other error occurred
                result.put("exists", false);
                result.put("message", "Error checking set: " + e.getMessage());
                result.put("error", e.getClass().getSimpleName());
            }
        }

        return result;
    }

    /**
     * Creates a set with multiple values in a single operation.
     *
     * <p>This method demonstrates how to create a set with multiple values
     * in a single update operation.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param setAttributeName The name of the set attribute
     * @param setValues The values to include in the set
```

```
     * @param isNumberSet Whether to create a number set (true) or string set
(false)
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse createSetWithValues(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String setAttributeName,
        Set<?> setValues,
        boolean isNumberSet) {

        AttributeValue setValue;

        if (isNumberSet) {
            // Convert numbers to strings for DynamoDB
            Set<String> stringValues = new HashSet<>();
            for (Object value : setValues) {
                if (value instanceof Number) {
                    stringValues.add(value.toString());
                } else {
                    throw new IllegalArgumentException("Values must be numbers for a
number set");
                }
            }

            setValue = AttributeValue.builder().ns(stringValues).build();
        } else {
            // Convert objects to strings for DynamoDB
            Set<String> stringValues = new HashSet<>();
            for (Object value : setValues) {
                stringValues.add(value.toString());
            }

            setValue = AttributeValue.builder().ss(stringValues).build();
        }

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #setAttr = :setValue")
            .expressionAttributeNames(Map.of("#setAttr", setAttributeName))
```

```
            .expressionAttributeValues(Map.of(":setValue", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Gets the current value of a set attribute.
     *
     * <p>Helper method to retrieve the current value of a set attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @param setAttributeName The name of the set attribute
     * @return The set attribute value or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static AttributeValue getSetAttribute(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, String setAttributeName) {

        // Define the get parameters
        GetItemRequest request = GetItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .projectionExpression(setAttributeName)
            .build();

        try {
            // Perform the get operation
            GetItemResponse response = dynamoDbClient.getItem(request);

            // Return the set attribute if it exists, otherwise null
            if (response.item() != null &&
response.item().containsKey(setAttributeName)) {
                return response.item().get(setAttributeName);
            }

            return null;
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
```

```
                .message("Failed to get set attribute: " + e.getMessage())
                .cause(e)
                .build();
        }
    }
```

使用集合运算的用法示例 AWS SDK for Java 2.x。

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating set operations in DynamoDB");

        try {
            // Example 1: Create a string set with multiple values
            System.out.println("\nExample 1: Creating a string set with multiple
values");
            Set<String> tags = new HashSet<>();
            tags.add("Electronics");
            tags.add("Gadget");
            tags.add("Smartphone");

            UpdateItemResponse createResponse = createSetWithValues(
                dynamoDbClient, tableName, key, "Tags", tags, false // Not a number
set
                );

            System.out.println("Created set attribute: " +
createResponse.attributes());

            // Example 2: Add values to a string set
            System.out.println("\nExample 2: Adding values to a string set");
            Set<String> additionalTags = new HashSet<>();
            additionalTags.add("Mobile");
            additionalTags.add("Wireless");

            UpdateItemResponse addResponse = addToStringSet(dynamoDbClient,
tableName, key, "Tags", additionalTags);
```

```
            System.out.println("Updated set attribute: " +
addResponse.attributes());

            // Example 3: Create a number set with multiple values
            System.out.println("\nExample 3: Creating a number set with multiple
values");
            Set<Number> ratings = new HashSet<>();
            ratings.add(4);
            ratings.add(5);
            ratings.add(4.5);

            UpdateItemResponse createNumberSetResponse = createSetWithValues(
                dynamoDbClient, tableName, key, "Ratings", ratings, true // Is a
number set
                );

            System.out.println("Created number set attribute: " +
createNumberSetResponse.attributes());

            // Example 4: Add values to a number set
            System.out.println("\nExample 4: Adding values to a number set");
            Set<Number> additionalRatings = new HashSet<>();
            additionalRatings.add(3.5);
            additionalRatings.add(4.2);

            UpdateItemResponse addNumberResponse =
                addToNumberSet(dynamoDbClient, tableName, key, "Ratings",
additionalRatings);

            System.out.println("Updated number set attribute: " +
addNumberResponse.attributes());

            // Example 5: Remove values from a set
            System.out.println("\nExample 5: Removing values from a set");
            Set<String> tagsToRemove = new HashSet<>();
            tagsToRemove.add("Gadget");

            UpdateItemResponse removeResponse = removeFromSet(
                dynamoDbClient, tableName, key, "Tags", tagsToRemove, false // Not a
number set
                );

            System.out.println("Updated set after removal: " +
removeResponse.attributes());
```

```
            // Example 6: Check if a value exists in a set
            System.out.println("\nExample 6: Checking if a value exists in a set");
            Map<String, Object> checkResult = checkIfValueInSet(dynamoDbClient,
tableName, key, "Tags", "Electronics");

            System.out.println("Check result: " + checkResult.get("message"));

            // Example 7: Get the current value of a set attribute
            System.out.println("\nExample 7: Getting the current value of a set
attribute");
            AttributeValue currentStringSet = getSetAttribute(dynamoDbClient,
tableName, key, "Tags");

            if (currentStringSet != null && currentStringSet.ss() != null) {
                System.out.println("Current string set values: " +
currentStringSet.ss());
            } else {
                System.out.println("String set attribute not found");
            }

            AttributeValue currentNumberSet = getSetAttribute(dynamoDbClient,
tableName, key, "Ratings");

            if (currentNumberSet != null && currentNumberSet.ns() != null) {
                System.out.println("Current number set values: " +
currentNumberSet.ns());
            } else {
                System.out.println("Number set attribute not found");
            }

            // Explain set operations
            System.out.println("\nKey points about DynamoDB set operations:");
            System.out.println(
                "1. DynamoDB supports three set types: string sets (SS), number sets
(NS), and binary sets (BS)");
            System.out.println("2. Sets can only contain elements of the same
type");
            System.out.println("3. Use ADD to add elements to a set");
            System.out.println("4. Use DELETE to remove elements from a set");
            System.out.println("5. Sets automatically remove duplicate values");
            System.out.println("6. Sets are unordered collections");
            System.out.println("7. Use the contains function to check if a value
exists in a set");
```

```
            System.out.println("8. You can create a set with multiple values in a
 single operation");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考UpdateItem中的。

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 语句来获取一批项目。
- 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。

适用于 Java 的 SDK 2.x

> **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。

```
public class ScenarioPartiQLBatch {
    public static void main(String[] args) throws IOException {
        String tableName = "MoviesPartiQBatch";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        System.out.println("Creating an Amazon DynamoDB table named " + tableName
            + " with a key named year and a sort key named title.");
        createTable(ddb, tableName);
```

```
        System.out.println("Adding multiple records into the " + tableName
            + " table using a batch command.");
        putRecordBatch(ddb);

        // Update multiple movies by using the BatchExecute statement.
        String title1 = "Star Wars";
        int year1 = 1977;
        String title2 = "Wizard of Oz";
        int year2 = 1939;

        System.out.println("Query two movies.");
        getBatch(ddb, tableName, title1, title2, year1, year2);

        System.out.println("Updating multiple records using a batch command.");
        updateTableItemBatch(ddb);

        System.out.println("Deleting multiple records using a batch command.");
        deleteItemBatch(ddb);

        System.out.println("Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static boolean getBatch(DynamoDbClient ddb, String tableName, String
title1, String title2, int year1, int year2) {
        String getBatch = "SELECT * FROM " + tableName + " WHERE title = ? AND year
= ?";

        List<BatchStatementRequest> statements = new ArrayList<>();
        statements.add(BatchStatementRequest.builder()
            .statement(getBatch)
            .parameters(AttributeValue.builder().s(title1).build(),
                AttributeValue.builder().n(String.valueOf(year1)).build())
            .build());
        statements.add(BatchStatementRequest.builder()
            .statement(getBatch)
            .parameters(AttributeValue.builder().s(title2).build(),
                AttributeValue.builder().n(String.valueOf(year2)).build())
            .build());

        BatchExecuteStatementRequest batchExecuteStatementRequest =
BatchExecuteStatementRequest.builder()
```

```
            .statements(statements)
            .build();

        try {
            BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchExecuteStatementRequest);
            if (!response.responses().isEmpty()) {
                response.responses().forEach(r -> {
                    System.out.println(r.item().get("title") + "\\t" +
r.item().get("year"));
                });
                return true;
            } else {
                System.out.println("Couldn't find either " + title1 + " or " +
title2 + ".");
                return false;
            }
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            return false;
        }
    }

    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();
```

```
        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE) // Sort
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)
            .billingMode(BillingMode.PAY_PER_REQUEST) //  DynamoDB automatically
scales based on traffic.
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse = dbWaiter
                .waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            String newTable = response.tableDescription().tableName();
            System.out.println("The " + newTable + " was successfully created.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void putRecordBatch(DynamoDbClient ddb) {
        String sqlStatement = "INSERT INTO MoviesPartiQBatch VALUE {'year':?,
'title' : ?, 'info' : ?}";
        try {
            // Create three movies to add to the Amazon DynamoDB table.
            // Set data for Movie 1.
            List<AttributeValue> parameters = new ArrayList<>();
```

```
            AttributeValue att1 = AttributeValue.builder()
                .n("1977")
                .build();

            AttributeValue att2 = AttributeValue.builder()
                .s("Star Wars")
                .build();

            AttributeValue att3 = AttributeValue.builder()
                .s("No Information")
                .build();

            parameters.add(att1);
            parameters.add(att2);
            parameters.add(att3);

            BatchStatementRequest statementRequestMovie1 =
BatchStatementRequest.builder()
                .statement(sqlStatement)
                .parameters(parameters)
                .build();

            // Set data for Movie 2.
            List<AttributeValue> parametersMovie2 = new ArrayList<>();
            AttributeValue attMovie2 = AttributeValue.builder()
                .n("1939")
                .build();

            AttributeValue attMovie2A = AttributeValue.builder()
                .s("Wizard of Oz")
                .build();

            AttributeValue attMovie2B = AttributeValue.builder()
                .s("No Information")
                .build();

            parametersMovie2.add(attMovie2);
            parametersMovie2.add(attMovie2A);
            parametersMovie2.add(attMovie2B);

            BatchStatementRequest statementRequestMovie2 =
BatchStatementRequest.builder()
                .statement(sqlStatement)
                .parameters(parametersMovie2)
```

```
                .build();

            // Set data for Movie 3.
            List<AttributeValue> parametersMovie3 = new ArrayList<>();
            AttributeValue attMovie3 = AttributeValue.builder()
                .n(String.valueOf("2022"))
                .build();

            AttributeValue attMovie3A = AttributeValue.builder()
                .s("My Movie 3")
                .build();

            AttributeValue attMovie3B = AttributeValue.builder()
                .s("No Information")
                .build();

            parametersMovie3.add(attMovie3);
            parametersMovie3.add(attMovie3A);
            parametersMovie3.add(attMovie3B);

            BatchStatementRequest statementRequestMovie3 =
BatchStatementRequest.builder()
                .statement(sqlStatement)
                .parameters(parametersMovie3)
                .build();

            // Add all three movies to the list.
            List<BatchStatementRequest> myBatchStatementList = new ArrayList<>();
            myBatchStatementList.add(statementRequestMovie1);
            myBatchStatementList.add(statementRequestMovie2);
            myBatchStatementList.add(statementRequestMovie3);

            BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
                .statements(myBatchStatementList)
                .build();

            BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchRequest);
            System.out.println("ExecuteStatement successful: " +
response.toString());
            System.out.println("Added new movies using a batch command.");

        } catch (DynamoDbException e) {
```

```
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void updateTableItemBatch(DynamoDbClient ddb) {
        String sqlStatement = "UPDATE MoviesPartiQBatch SET info = 'directors\":
[\"Merian C. Cooper\",\"Ernest B. Schoedsack' where year=? and title=?";
        List<AttributeValue> parametersRec1 = new ArrayList<>();

        // Update three records.
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("My Movie 1")
            .build();

        parametersRec1.add(att1);
        parametersRec1.add(att2);

        BatchStatementRequest statementRequestRec1 = BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec1)
            .build();

        // Update record 2.
        List<AttributeValue> parametersRec2 = new ArrayList<>();
        AttributeValue attRec2 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attRec2a = AttributeValue.builder()
            .s("My Movie 2")
            .build();

        parametersRec2.add(attRec2);
        parametersRec2.add(attRec2a);
        BatchStatementRequest statementRequestRec2 = BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec2)
            .build();
```

```
        // Update record 3.
        List<AttributeValue> parametersRec3 = new ArrayList<>();
        AttributeValue attRec3 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attRec3a = AttributeValue.builder()
            .s("My Movie 3")
            .build();

        parametersRec3.add(attRec3);
        parametersRec3.add(attRec3a);
        BatchStatementRequest statementRequestRec3 = BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec3)
            .build();

        // Add all three movies to the list.
        List<BatchStatementRequest> myBatchStatementList = new ArrayList<>();
        myBatchStatementList.add(statementRequestRec1);
        myBatchStatementList.add(statementRequestRec2);
        myBatchStatementList.add(statementRequestRec3);

        BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
            .statements(myBatchStatementList)
            .build();

        try {
            BatchExecuteStatementResponse response =
ddb.batchExecuteStatement(batchRequest);
            System.out.println("ExecuteStatement successful: " +
response.toString());
            System.out.println("Updated three movies using a batch command.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println("Item was updated!");
    }

    public static void deleteItemBatch(DynamoDbClient ddb) {
```

```
        String sqlStatement = "DELETE FROM MoviesPartiQBatch WHERE year = ? and
title=?";
        List<AttributeValue> parametersRec1 = new ArrayList<>();

        // Specify three records to delete.
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("My Movie 1")
            .build();

        parametersRec1.add(att1);
        parametersRec1.add(att2);

        BatchStatementRequest statementRequestRec1 = BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec1)
            .build();

        // Specify record 2.
        List<AttributeValue> parametersRec2 = new ArrayList<>();
        AttributeValue attRec2 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();

        AttributeValue attRec2a = AttributeValue.builder()
            .s("My Movie 2")
            .build();

        parametersRec2.add(attRec2);
        parametersRec2.add(attRec2a);
        BatchStatementRequest statementRequestRec2 = BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec2)
            .build();

        // Specify record 3.
        List<AttributeValue> parametersRec3 = new ArrayList<>();
        AttributeValue attRec3 = AttributeValue.builder()
            .n(String.valueOf("2022"))
            .build();
```

```
        AttributeValue attRec3a = AttributeValue.builder()
            .s("My Movie 3")
            .build();

        parametersRec3.add(attRec3);
        parametersRec3.add(attRec3a);

        BatchStatementRequest statementRequestRec3 = BatchStatementRequest.builder()
            .statement(sqlStatement)
            .parameters(parametersRec3)
            .build();

        // Add all three movies to the list.
        List<BatchStatementRequest> myBatchStatementList = new ArrayList<>();
        myBatchStatementList.add(statementRequestRec1);
        myBatchStatementList.add(statementRequestRec2);
        myBatchStatementList.add(statementRequestRec3);

        BatchExecuteStatementRequest batchRequest =
BatchExecuteStatementRequest.builder()
            .statements(myBatchStatementList)
            .build();

        try {
            ddb.batchExecuteStatement(batchRequest);
            System.out.println("Deleted three movies using a batch command.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {
        DeleteTableRequest request = DeleteTableRequest.builder()
            .tableName(tableName)
            .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
```

```
        }
        System.out.println(tableName + " was successfully deleted!");
    }

    private static ExecuteStatementResponse executeStatementRequest(DynamoDbClient
 ddb, String statement,

 List<AttributeValue> parameters) {
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
            .statement(statement)
            .parameters(parameters)
            .build();

        return ddb.executeStatement(request);
    }
}
```

- 有关 API 的详细信息，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 语句来获取项目。
- 通过运行 INSERT 语句来添加项目。
- 通过运行 UPDATE 语句来更新项目。
- 通过运行 DELETE 语句来删除项目。

适用于 Java 的 SDK 2.x

> **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。

```
public class ScenarioPartiQ {
    public static void main(String[] args) throws IOException {
        String fileName = "../../../resources/sample_files/movies.json";
```

```
        String tableName = "MoviesPartiQ";
        Region region = Region.US_EAST_1;
        DynamoDbClient ddb = DynamoDbClient.builder()
            .region(region)
            .build();

        System.out.println(
            "******* Creating an Amazon DynamoDB table named MoviesPartiQ with a key
named year and a sort key named title.");
        createTable(ddb, tableName);

        System.out.println("Loading data into the MoviesPartiQ table.");
        loadData(ddb, fileName);

        System.out.println("Getting data from the MoviesPartiQ table.");
        getItem(ddb);

        System.out.println("Putting a record into the MoviesPartiQ table.");
        putRecord(ddb);

        System.out.println("Updating a record.");
        updateTableItem(ddb);

        System.out.println("Querying the movies released in 2013.");
        queryTable(ddb);

        System.out.println("Deleting the Amazon DynamoDB table.");
        deleteDynamoDBTable(ddb, tableName);
        ddb.close();
    }

    public static void createTable(DynamoDbClient ddb, String tableName) {
        DynamoDbWaiter dbWaiter = ddb.waiter();
        ArrayList<AttributeDefinition> attributeDefinitions = new ArrayList<>();

        // Define attributes.
        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("year")
            .attributeType("N")
            .build());

        attributeDefinitions.add(AttributeDefinition.builder()
            .attributeName("title")
            .attributeType("S")
```

```
            .build());

        ArrayList<KeySchemaElement> tableKey = new ArrayList<>();
        KeySchemaElement key = KeySchemaElement.builder()
            .attributeName("year")
            .keyType(KeyType.HASH)
            .build();

        KeySchemaElement key2 = KeySchemaElement.builder()
            .attributeName("title")
            .keyType(KeyType.RANGE) // Sort
            .build();

        // Add KeySchemaElement objects to the list.
        tableKey.add(key);
        tableKey.add(key2);

        CreateTableRequest request = CreateTableRequest.builder()
            .keySchema(tableKey)
            .billingMode(BillingMode.PAY_PER_REQUEST) //Scales based on traffic.
            .attributeDefinitions(attributeDefinitions)
            .tableName(tableName)
            .build();

        try {
            CreateTableResponse response = ddb.createTable(request);
            DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            // Wait until the Amazon DynamoDB table is created.
            WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            String newTable = response.tableDescription().tableName();
            System.out.println("The " + newTable + " was successfully created.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Load data into the table.
```

```
   public static void loadData(DynamoDbClient ddb, String fileName) throws
IOException {

       String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?,
'title' : ?, 'info' : ?}";
       JsonParser parser = new JsonFactory().createParser(new File(fileName));
       com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
       Iterator<JsonNode> iter = rootNode.iterator();
       ObjectNode currentNode;
       int t = 0;
       List<AttributeValue> parameters = new ArrayList<>();
       while (iter.hasNext()) {

           // Add 200 movies to the table.
           if (t == 200)
               break;
           currentNode = (ObjectNode) iter.next();

           int year = currentNode.path("year").asInt();
           String title = currentNode.path("title").asText();
           String info = currentNode.path("info").toString();

           AttributeValue att1 = AttributeValue.builder()
               .n(String.valueOf(year))
               .build();

           AttributeValue att2 = AttributeValue.builder()
               .s(title)
               .build();

           AttributeValue att3 = AttributeValue.builder()
               .s(info)
               .build();

           parameters.add(att1);
           parameters.add(att2);
           parameters.add(att3);

           // Insert the movie into the Amazon DynamoDB table.
           executeStatementRequest(ddb, sqlStatement, parameters);
           System.out.println("Added Movie " + title);

           parameters.remove(att1);
```

```
            parameters.remove(att2);
            parameters.remove(att3);
            t++;
        }
    }

    public static void getItem(DynamoDbClient ddb) {

        String sqlStatement = "SELECT * FROM MoviesPartiQ where year=? and title=?";
        List<AttributeValue> parameters = new ArrayList<>();
        AttributeValue att1 = AttributeValue.builder()
            .n("2012")
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("The Perks of Being a Wallflower")
            .build();

        parameters.add(att1);
        parameters.add(att2);

        try {
            ExecuteStatementResponse response = executeStatementRequest(ddb,
sqlStatement, parameters);
            System.out.println("ExecuteStatement successful: " +
response.toString());

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void putRecord(DynamoDbClient ddb) {

        String sqlStatement = "INSERT INTO MoviesPartiQ VALUE {'year':?,
'title' : ?, 'info' : ?}";
        try {
            List<AttributeValue> parameters = new ArrayList<>();

            AttributeValue att1 = AttributeValue.builder()
                .n(String.valueOf("2020"))
                .build();
```

```
            AttributeValue att2 = AttributeValue.builder()
                .s("My Movie")
                .build();

            AttributeValue att3 = AttributeValue.builder()
                .s("No Information")
                .build();

            parameters.add(att1);
            parameters.add(att2);
            parameters.add(att3);

            executeStatementRequest(ddb, sqlStatement, parameters);
            System.out.println("Added new movie.");

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void updateTableItem(DynamoDbClient ddb) {

        String sqlStatement = "UPDATE MoviesPartiQ SET info = 'directors\":[\"Merian
C. Cooper\",\"Ernest B. Schoedsack' where year=? and title=?";
        List<AttributeValue> parameters = new ArrayList<>();
        AttributeValue att1 = AttributeValue.builder()
            .n(String.valueOf("2013"))
            .build();

        AttributeValue att2 = AttributeValue.builder()
            .s("The East")
            .build();

        parameters.add(att1);
        parameters.add(att2);

        try {
            executeStatementRequest(ddb, sqlStatement, parameters);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
```

```
        System.out.println("Item was updated!");
    }

    // Query the table where the year is 2013.
    public static void queryTable(DynamoDbClient ddb) {
        String sqlStatement = "SELECT * FROM MoviesPartiQ where year = ? ORDER BY
year";
        try {

            List<AttributeValue> parameters = new ArrayList<>();
            AttributeValue att1 = AttributeValue.builder()
                .n(String.valueOf("2013"))
                .build();
            parameters.add(att1);

            // Get items in the table and write out the ID value.
            ExecuteStatementResponse response = executeStatementRequest(ddb,
sqlStatement, parameters);
            System.out.println("ExecuteStatement successful: " +
response.toString());

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteDynamoDBTable(DynamoDbClient ddb, String tableName) {

        DeleteTableRequest request = DeleteTableRequest.builder()
            .tableName(tableName)
            .build();

        try {
            ddb.deleteTable(request);

        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        System.out.println(tableName + " was successfully deleted!");
    }
```

```
    private static ExecuteStatementResponse executeStatementRequest(DynamoDbClient
 ddb, String statement,

 List<AttributeValue> parameters) {
        ExecuteStatementRequest request = ExecuteStatementRequest.builder()
            .statement(statement)
            .parameters(parameters)
            .build();

        return ddb.executeStatement(request);
    }

    private static void processResults(ExecuteStatementResponse
 executeStatementResult) {
        System.out.println("ExecuteStatement successful: " +
 executeStatementResult.toString());
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ExecuteStatement中的。

使用全局二级索引查询表

以下代码示例演示如何使用全局二级索引来查询表。

- 使用 DynamoDB 表的主键查询该表。
- 查询全局二级索引（GSI）以获取其它访问模式。
- 比较表查询和 GSI 查询。

适用于 Java 的 SDK 2.x

使用 DynamoDB 表的主键和全局二级索引 (GSI) 查询。 AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
```

```
import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryTable(
        final String tableName, final String partitionKeyName, final String
 partitionKeyValue) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query on base table successful. Found " +
 response.count() + " items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
 found.\n", tableName);
            throw new DynamoDbQueryException("Table not found: " + tableName, e);
        } catch (DynamoDbException e) {
            System.err.println("Error querying base table: " + e.getMessage());
            throw new DynamoDbQueryException("Failed to execute query on base
 table", e);
```

```
        }
    }

    /**
     * Queries a DynamoDB Global Secondary Index (GSI) by partition key.
     *
     * @param tableName         The name of the DynamoDB table
     * @param indexName         The name of the GSI
     * @param partitionKeyName  The name of the GSI partition key attribute
     * @param partitionKeyValue The value of the GSI partition key to query
     * @return The query response from DynamoDB
     * @throws ResourceNotFoundException if the table or index doesn't exist
     * @throws DynamoDbException if the query fails
     */
    public QueryResponse queryGlobalSecondaryIndex(
        final String tableName, final String indexName, final String
partitionKeyName, final String partitionKeyValue) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Index name", indexName);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_IK,
partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_IK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .indexName(indexName)
            .keyConditionExpression(GSI_KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
```

```
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query on GSI successful. Found " + response.count()
+ " items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format(
                "Error: The Amazon DynamoDB table \"%s\" or index \"%s\" can't be
found.\n", tableName, indexName);
            throw new DynamoDbQueryException("Table or index not found: " +
tableName + "/" + indexName, e);
        } catch (DynamoDbException e) {
            System.err.println("Error querying GSI: " + e.getMessage());
            throw new DynamoDbQueryException("Failed to execute query on GSI", e);
        }
    }
```

比较直接查询表和使用查询 GSI AWS SDK for Java 2.x。

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <basePartitionKeyName> <basePartitionKeyValue>
<gsiName> <gsiPartitionKeyName> <gsiPartitionKeyValue> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    basePartitionKeyName - The name of the base table partition key
attribute.
                    basePartitionKeyValue - The value of the base table partition
key to query.
                    gsiName - The name of the Global Secondary Index.
                    gsiPartitionKeyName - The name of the GSI partition key
attribute.
                    gsiPartitionKeyValue - The value of the GSI partition key to
query.
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 6) {
            System.out.println(usage);
            System.exit(1);
```

```
        }

        final String tableName = args[0];
        final String basePartitionKeyName = args[1];
        final String basePartitionKeyValue = args[2];
        final String gsiName = args[3];
        final String gsiPartitionKeyName = args[4];
        final String gsiPartitionKeyValue = args[5];
        final Region region = args.length > 6 ? Region.of(args[6]) :
Region.US_EAST_1;

        try (DynamoDbClient ddb = DynamoDbClient.builder().region(region).build()) {
            final QueryTableAndGSI queryHelper = new QueryTableAndGSI(ddb);

            // Query the base table
            System.out.println("Querying base table where " + basePartitionKeyName +
" = " + basePartitionKeyValue);
            final QueryResponse tableResponse =
                queryHelper.queryTable(tableName, basePartitionKeyName,
basePartitionKeyValue);

            System.out.println("Found " + tableResponse.count() + " items in base
table:");
            tableResponse.items().forEach(item -> System.out.println(item));

            // Query the GSI
            System.out.println(
                "\nQuerying GSI '" + gsiName + "' where " + gsiPartitionKeyName + "
= " + gsiPartitionKeyValue);
            final QueryResponse gsiResponse =
                queryHelper.queryGlobalSecondaryIndex(tableName, gsiName,
gsiPartitionKeyName, gsiPartitionKeyValue);

            System.out.println("Found " + gsiResponse.count() + " items in GSI:");
            gsiResponse.items().forEach(item -> System.out.println(item));

            // Explain the differences between querying a table and a GSI
            System.out.println("\nKey differences between querying a table and a
GSI:");
            System.out.println("1. When querying a GSI, you must specify the
indexName parameter");
            System.out.println("2. GSIs may not contain all attributes from the base
table (projection)");
```

```
            System.out.println("3. GSIs consume read capacity units from the GSI's
capacity, not the base table's");
            System.out.println("4. GSIs may have eventually consistent data (cannot
use ConsistentRead=true)");

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table or index not found: " + e.getMessage());
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 Query。

使用 begins_with 条件查询表

以下代码示例演示如何使用 begins_with 条件来查询表。

- 在键条件表达式中使用 begins_with 函数。
- 根据排序键中的前缀模式筛选项目。

适用于 Java 的 SDK 2.x

使用 AWS SDK for Java 2.x通过排序键上的 begins_with 条件查询 DynamoDB 表。

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
```

```
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithBeginsWithCondition(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String sortKeyName,
        final String sortKeyPrefix) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Sort key name", sortKeyName);
        CodeSampleUtils.validateStringParameter("Sort key prefix", sortKeyPrefix);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_SK, sortKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_SK_PREFIX,
            AttributeValue.builder().s(sortKeyPrefix).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query with begins_with condition successful.
 Found {0} items", response.count());
```

```
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with begins_with condition",
e);
            throw e;
        }
    }
```

演示如何通过 AWS SDK for Java 2.x将 begins_with 与不同的前缀长度结合使用。

```
    public static void main(String[] args) {
        try {
            CodeSampleUtils.BeginsWithQueryConfig config =
CodeSampleUtils.BeginsWithQueryConfig.fromArgs(args);
            LOGGER.log(Level.INFO, "Querying items where {0} = {1} and {2} begins
with ''{3}''", new Object[] {
                config.getPartitionKeyName(),
                config.getPartitionKeyValue(),
                config.getSortKeyName(),
                config.getSortKeyPrefix()
            });

            // Using the builder pattern to create and execute the query
            final QueryResponse response = new BeginsWithQueryBuilder()
                .withTableName(config.getTableName())
                .withPartitionKeyName(config.getPartitionKeyName())
                .withPartitionKeyValue(config.getPartitionKeyValue())
                .withSortKeyName(config.getSortKeyName())
                .withSortKeyPrefix(config.getSortKeyPrefix())
                .withRegion(config.getRegion())
                .execute();

            // Process the results
            LOGGER.log(Level.INFO, "Found {0} items:", response.count());
            response.items().forEach(item -> LOGGER.info(item.toString()));

            // Demonstrate with a different prefix
            if (!config.getSortKeyPrefix().isEmpty()) {
                String shorterPrefix = config.getSortKeyPrefix()
```

```
                    .substring(0, Math.max(1, config.getSortKeyPrefix().length() /
2));
                LOGGER.log(Level.INFO, "\nNow querying with a shorter prefix:
''{0}''", shorterPrefix);

                final QueryResponse response2 = new BeginsWithQueryBuilder()
                    .withTableName(config.getTableName())
                    .withPartitionKeyName(config.getPartitionKeyName())
                    .withPartitionKeyValue(config.getPartitionKeyValue())
                    .withSortKeyName(config.getSortKeyName())
                    .withSortKeyPrefix(shorterPrefix)
                    .withRegion(config.getRegion())
                    .execute();

                LOGGER.log(Level.INFO, "Found {0} items with shorter prefix:",
response2.count());
                response2.items().forEach(item -> LOGGER.info(item.toString()));
            }
        } catch (IllegalArgumentException e) {
            LOGGER.log(Level.SEVERE, "Invalid input: {0}", e.getMessage());
            printUsage();
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found", e);
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "DynamoDB error", e);
        } catch (Exception e) {
            LOGGER.log(Level.SEVERE, "Unexpected error", e);
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 Query。

使用日期范围查询表

以下代码示例演示如何使用排序键中的日期范围来查询表。

- 查询特定日期范围内的项目。
- 对日期格式的排序键使用比较运算符。

适用于 Java 的 SDK 2.x

在 DynamoDB 表中查询 DynamoDB 表中是否有某个日期范围内的项目。 AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.time.LocalDate;
import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithDateRange(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final LocalDate startDate,
        final LocalDate endDate) {

        // Focus on query logic, assuming parameters are valid
        if (startDate == null || endDate == null) {
            throw new IllegalArgumentException("Start date and end date cannot be
 null");
        }

        if (endDate.isBefore(startDate)) {
            throw new IllegalArgumentException("End date must be after start date");
        }

        // Format dates as ISO strings for DynamoDB (using just the date part)
        final String formattedStartDate = startDate.toString();
        final String formattedEndDate = endDate.toString();

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);
```

```
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_SK, dateKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_START_DATE,
            AttributeValue.builder().s(formattedStartDate).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_END_DATE,
            AttributeValue.builder().s(formattedEndDate).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query by date range successful. Found {0}
items", response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying by date range: {0}",
e.getMessage());
            throw e;
        }
    }
```

演示如何使用日期范围筛选来查询 DynamoDB 表。

```
    public static void main(String[] args) {
        final String usage =
```

```
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue> <dateKeyName>
<startDate> <endDate> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    dateKeyName - The name of the date attribute to filter on.
                    startDate - The start date for the range query (YYYY-MM-DD).
                    endDate - The end date for the range query (YYYY-MM-DD).
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 6) {
            System.out.println(usage);
            System.exit(1);
        }

        try {
            // Parse command line arguments into a config object
            CodeSampleUtils.DateRangeQueryConfig config =
CodeSampleUtils.DateRangeQueryConfig.fromArgs(args);

            LOGGER.log(
                Level.INFO, "Querying items from {0} to {1}", new Object[]
{config.getStartDate(), config.getEndDate()
                });

            // Using the builder pattern to create and execute the query
            final QueryResponse response = new DateRangeQueryBuilder()
                .withTableName(config.getTableName())
                .withPartitionKeyName(config.getPartitionKeyName())
                .withPartitionKeyValue(config.getPartitionKeyValue())
                .withDateKeyName(config.getDateKeyName())
                .withStartDate(config.getStartDate())
                .withEndDate(config.getEndDate())
                .withRegion(config.getRegion())
                .execute();

            // Process the results
            LOGGER.log(Level.INFO, "Found {0} items:", response.count());
            response.items().forEach(item -> {
```

```
                LOGGER.info(item.toString());

                // Extract and display the date attribute for clarity
                if (item.containsKey(config.getDateKeyName())) {
                    LOGGER.log(
                        Level.INFO,
                        "  Date attribute: {0}",
                        item.get(config.getDateKeyName()).s());
                }
            });

            // Demonstrate with a different date range
            LocalDate narrowerStartDate = config.getStartDate().plusDays(1);
            LocalDate narrowerEndDate = config.getEndDate().minusDays(1);

            if (!narrowerStartDate.isAfter(narrowerEndDate)) {
                LOGGER.log(Level.INFO, "\nNow querying with a narrower date range:
{0} to {1}", new Object[] {
                    narrowerStartDate, narrowerEndDate
                });

                final QueryResponse response2 = new DateRangeQueryBuilder()
                    .withTableName(config.getTableName())
                    .withPartitionKeyName(config.getPartitionKeyName())
                    .withPartitionKeyValue(config.getPartitionKeyValue())
                    .withDateKeyName(config.getDateKeyName())
                    .withStartDate(narrowerStartDate)
                    .withEndDate(narrowerEndDate)
                    .withRegion(config.getRegion())
                    .execute();

                LOGGER.log(Level.INFO, "Found {0} items with narrower date range:",
response2.count());
                response2.items().forEach(item -> LOGGER.info(item.toString()));
            }

            LOGGER.info("\nNote: When storing dates in DynamoDB:");
            LOGGER.info("1. Use ISO format (YYYY-MM-DD) for lexicographical
ordering");
            LOGGER.info("2. Use the BETWEEN operator for inclusive date range
queries");
            LOGGER.info("3. Consider using ISO-8601 format for timestamps with time
components");
```

```
        } catch (IllegalArgumentException e) {
            LOGGER.log(Level.SEVERE, "Invalid input: {0}", e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", e.getMessage());
            System.exit(1);
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "DynamoDB error: {0}", e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            LOGGER.log(Level.SEVERE, "Unexpected error: {0}", e.getMessage());
            System.exit(1);
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 Query。

使用复杂的筛选表达式查询表

以下代码示例演示如何使用复杂的筛选表达式来查询表。

- 将复杂的筛选表达式应用于查询结果。
- 使用逻辑运算符组合条件。
- 根据非键属性筛选项目。

适用于 Java 的 SDK 2.x

使用复杂筛选表达式查询 DynamoDB 表。 AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
```

```
import java.util.logging.Logger;

    public QueryResponse queryWithComplexFilter(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String statusAttrName,
        final String activeStatus,
        final String pendingStatus,
        final String priceAttrName,
        final double minPrice,
        final double maxPrice,
        final String categoryAttrName) {

        // Validate parameters
        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Status attribute name",
 statusAttrName);
        CodeSampleUtils.validateStringParameter("Active status", activeStatus);
        CodeSampleUtils.validateStringParameter("Pending status", pendingStatus);
        CodeSampleUtils.validateStringParameter("Price attribute name",
 priceAttrName);
        CodeSampleUtils.validateStringParameter("Category attribute name",
 categoryAttrName);
        CodeSampleUtils.validateNumericRange("Minimum price", minPrice, 0.0,
 Double.MAX_VALUE);
        CodeSampleUtils.validateNumericRange("Maximum price", maxPrice, minPrice,
 Double.MAX_VALUE);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put("#pk", partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_STATUS,
 statusAttrName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PRICE,
 priceAttrName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_CATEGORY,
 categoryAttrName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
```

```
            ":pkValue", AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_ACTIVE,
            AttributeValue.builder().s(activeStatus).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PENDING,
            AttributeValue.builder().s(pendingStatus).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_MIN_PRICE,
            AttributeValue.builder().n(String.valueOf(minPrice)).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_MAX_PRICE,
            AttributeValue.builder().n(String.valueOf(maxPrice)).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(FILTER_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        return dynamoDbClient.query(queryRequest);
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 Query。

使用动态筛选表达式查询表

以下代码示例演示如何使用动态筛选表达式来查询表。

- 在运行时动态构建筛选表达式。
- 根据用户输入或应用程序状态构造筛选条件。
- 有条件地添加或移除筛选条件。

适用于 Java 的 SDK 2.x

使用动态构造的筛选表达式查询 DynamoDB 表。 AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;

    public static QueryResponse queryWithDynamicFilter(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final Map<String, Object> filterCriteria,
        final Region region,
        final DynamoDbClient dynamoDbClient) {

        validateParameters(tableName, partitionKeyName, partitionKeyValue,
 filterCriteria);

        DynamoDbClient ddbClient = dynamoDbClient;
        boolean shouldClose = false;

        try {
            if (ddbClient == null) {
                ddbClient = createClient(region);
                shouldClose = true;
            }

            final QueryWithDynamicFilter queryHelper = new
 QueryWithDynamicFilter(ddbClient);
            return queryHelper.queryWithDynamicFilter(tableName, partitionKeyName,
 partitionKeyValue, filterCriteria);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Failed to execute dynamic filter query: " +
 e.getMessage());
            throw e;
        } catch (Exception e) {
```

```
            System.err.println("Unexpected error during query: " + e.getMessage());
            throw e;
        } finally {
            if (shouldClose && ddbClient != null) {
                ddbClient.close();
            }
        }
    }
```

演示如何使用动态过滤器表达式 AWS SDK for Java 2.x。

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue>
<filterAttrName> <filterAttrValue> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    filterAttrName - The name of the attribute to filter on.
                    filterAttrValue - The value to filter by.
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 5) {
            System.out.println(usage);
            System.exit(1);
        }

        final String tableName = args[0];
        final String partitionKeyName = args[1];
        final String partitionKeyValue = args[2];
        final String filterAttrName = args[3];
        final String filterAttrValue = args[4];
        final Region region = args.length > 5 ? Region.of(args[5]) :
Region.US_EAST_1;

        System.out.println("Querying items with dynamic filter: " + filterAttrName +
" = " + filterAttrValue);
```

```
        try {
            // Using the builder pattern to create and execute the query
            final QueryResponse response = new DynamicFilterQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withFilterCriterion(filterAttrName, filterAttrValue)
                .withRegion(region)
                .execute();

            // Process the results
            System.out.println("Found " + response.count() + " items:");
            response.items().forEach(item -> System.out.println(item));

            // Demonstrate multiple filter criteria
            System.out.println("\nNow querying with multiple filter criteria:");

            Map<String, Object> multipleFilters = new HashMap<>();
            multipleFilters.put(filterAttrName, filterAttrValue);
            multipleFilters.put("status", "active");

            final QueryResponse multiFilterResponse = new
DynamicFilterQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withFilterCriteria(multipleFilters)
                .withRegion(region)
                .execute();

            System.out.println("Found " + multiFilterResponse.count() + " items with
multiple filters:");
            multiFilterResponse.items().forEach(item -> System.out.println(item));

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
```

```
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 Query。

使用筛选表达式和限制查询表

以下代码示例演示如何使用筛选表达式和限制来查询表。

- 将筛选表达式应用于查询结果，但对评估的项目有限制。
- 了解限制如何影响筛选的查询结果。
- 控制查询中处理的最大项目数。

适用于 Java 的 SDK 2.x

使用筛选表达式查询 DynamoDB 表，并使用限制。 AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithFilterAndLimit(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String filterAttrName,
        final String filterAttrValue,
        final int limit) {
```

```
        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Filter attribute name",
filterAttrName);
        CodeSampleUtils.validateStringParameter("Filter attribute value",
filterAttrValue);
        CodeSampleUtils.validatePositiveInteger("Limit", limit);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_FILTER,
filterAttrName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_FILTER,
            AttributeValue.builder().s(filterAttrValue).build());

        // Create the filter expression
        final String filterExpression = "#filterAttr = :filterValue";

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(filterExpression)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .limit(limit)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query with filter and limit successful. Found
{0} items", response.count());
            LOGGER.log(
```

```
                Level.INFO, "ScannedCount: {0} (total items evaluated before
filtering)", response.scannedCount());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with filter and limit: {0}",
e.getMessage());
            throw e;
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 Query。

使用嵌套属性查询表

以下代码示例演示如何使用嵌套属性来查询表。

- 按 DynamoDB 项目中的嵌套属性进行访问和筛选。
- 使用文档路径表达式来引用嵌套元素。

适用于 Java 的 SDK 2.x

使用查询带有嵌套属性的 DynamoDB 表。 AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryWithNestedAttributes(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
```

```
        final String nestedPath,
        final String nestedAttr,
        final String nestedValue) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Nested path", nestedPath);
        CodeSampleUtils.validateStringParameter("Nested attribute", nestedAttr);
        CodeSampleUtils.validateStringParameter("Nested value", nestedValue);

        // Split the nested path into components
        final String[] pathComponents = nestedPath.split("\\.");

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);

        // Build the nested attribute reference using document path notation
        final StringBuilder nestedAttributeRef = new StringBuilder();
        for (int i = 0; i < pathComponents.length; i++) {
            final String aliasName = "#n" + i;
            expressionAttributeNames.put(aliasName, pathComponents[i]);

            if (i > 0) {
                nestedAttributeRef.append(".");
            }
            nestedAttributeRef.append(aliasName);
        }

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_NESTED,
            AttributeValue.builder().s(nestedValue).build());

        // Create the filter expression using the nested attribute reference
        final String filterExpression = nestedAttributeRef + " = :nestedValue";

        // Create the query request
```

```
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(filterExpression)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query with nested attribute filter successful. Found
" + response.count() + " items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with nested attribute filter: " +
e.getMessage());
            throw e;
        }
    }
```

演示如何通过嵌套属性查询 DynamoDB 表。

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue> <nestedPath>
<nestedAttr> <nestedValue> [region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    nestedPath - The path to the nested map attribute (e.g.,
"address").
                    nestedAttr - The name of the nested attribute (e.g., "city").
                    nestedValue - The value to filter by (e.g., "Seattle").
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
```

```
                """;

        if (args.length < 6) {
            System.out.println(usage);
            System.exit(1);
        }

        final String tableName = args[0];
        final String partitionKeyName = args[1];
        final String partitionKeyValue = args[2];
        final String nestedPath = args[3];
        final String nestedAttr = args[4];
        final String nestedValue = args[5];
        final Region region = args.length > 6 ? Region.of(args[6]) :
Region.US_EAST_1;

        System.out.println("Querying items where " + partitionKeyName + " = " +
partitionKeyValue + " and " + nestedPath
            + "." + nestedAttr + " = " + nestedValue);

        try {
            // Using the builder pattern to create and execute the query
            final QueryResponse response = new NestedAttributeQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withNestedPath(nestedPath)
                .withNestedAttribute(nestedAttr)
                .withNestedValue(nestedValue)
                .withRegion(region)
                .execute();

            // Process the results
            System.out.println("Found " + response.count() + " items:");
            response.items().forEach(item -> {
                System.out.println(item);

                // Extract and display the nested attribute for clarity
                if (item.containsKey(nestedPath) && item.get(nestedPath).hasM()) {
                    Map<String, AttributeValue> nestedMap =
item.get(nestedPath).m();
                    if (nestedMap.containsKey(nestedAttr)) {
                        System.out.println("  Nested attribute " + nestedPath + "."
+ nestedAttr + ": "
```

```
                            + formatAttributeValue(nestedMap.get(nestedAttr)));
                    }
                }
            });

            System.out.println("\nNote: When working with nested attributes in
DynamoDB:");
            System.out.println("1. Use dot notation in filter expressions to access
nested attributes");
            System.out.println("2. Use expression attribute names for each component
of the path");
            System.out.println("3. Check if the nested attribute exists before
accessing it");

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 Query。

通过分区查询表

以下代码示例演示如何使用分区来查询表。

- 为 DynamoDB 查询结果实现分页。
- 使用检 LastEvaluatedKey 索后续页面。
- 使用 Limit 参数控制每页项目数。

适用于 Java 的 SDK 2.x

使用分页查询 DynamoDB 表。 AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    public List<Map<String, AttributeValue>> queryWithPagination(
        final String tableName, final String partitionKeyName, final String
 partitionKeyValue, final int pageSize) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validatePositiveInteger("Page size", pageSize);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        QueryRequest.Builder queryRequestBuilder = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .limit(pageSize);
```

```
        // List to store all items from all pages
        final List<Map<String, AttributeValue>> allItems = new ArrayList<>();

        // Map to store the last evaluated key for pagination
        Map<String, AttributeValue> lastEvaluatedKey = null;
        int pageNumber = 1;

        try {
            do {
                // If we have a last evaluated key, use it for the next page
                if (lastEvaluatedKey != null) {
                    queryRequestBuilder.exclusiveStartKey(lastEvaluatedKey);
                }

                // Execute the query
                final QueryResponse response =
dynamoDbClient.query(queryRequestBuilder.build());

                // Process the current page of results
                final List<Map<String, AttributeValue>> pageItems =
response.items();
                allItems.addAll(pageItems);

                // Get the last evaluated key for the next page
                lastEvaluatedKey = response.lastEvaluatedKey();
                if (lastEvaluatedKey != null && lastEvaluatedKey.isEmpty()) {
                    lastEvaluatedKey = null;
                }

                System.out.println("Page " + pageNumber + ": Retrieved " +
pageItems.size() + " items (Running total: "
                    + allItems.size() + ")");

                pageNumber++;

            } while (lastEvaluatedKey != null);

            System.out.println("Query with pagination complete. Retrieved a total of
" + allItems.size()
                + " items across " + (pageNumber - 1) + " pages");

            return allItems;
        } catch (ResourceNotFoundException e) {
```

```
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with pagination: " + e.getMessage());
            throw e;
        }
    }
```

演示如何使用分页查询 DynamoDB 表。

```
    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <tableName> <partitionKeyName> <partitionKeyValue> [pageSize]
[region]
                Where:
                    tableName - The Amazon DynamoDB table to query.
                    partitionKeyName - The name of the partition key attribute.
                    partitionKeyValue - The value of the partition key to query.
                    pageSize (optional) - The maximum number of items to return per
page. (Default: 10)
                    region (optional) - The AWS region where the table exists.
(Default: us-east-1)
                """;

        if (args.length < 3) {
            System.out.println(usage);
            System.exit(1);
        }

        final String tableName = args[0];
        final String partitionKeyName = args[1];
        final String partitionKeyValue = args[2];
        final int pageSize = args.length > 3 ? Integer.parseInt(args[3]) : 10;
        final Region region = args.length > 4 ? Region.of(args[4]) :
Region.US_EAST_1;

        System.out.println("Querying items with pagination (page size: " + pageSize
+ ")");
```

```
        try {
            // Using the builder pattern to create and execute the query
            final List<Map<String, AttributeValue>> allItems = new
PaginationQueryBuilder()
                .withTableName(tableName)
                .withPartitionKeyName(partitionKeyName)
                .withPartitionKeyValue(partitionKeyValue)
                .withPageSize(pageSize)
                .withRegion(region)
                .executeWithPagination();

            // Process the results
            System.out.println("\nSummary: Retrieved a total of " + allItems.size()
+ " items");

            // Display the first few items as a sample
            final int sampleSize = Math.min(5, allItems.size());
            if (sampleSize > 0) {
                System.out.println("\nSample of retrieved items (first " +
sampleSize + "):");
                for (int i = 0; i < sampleSize; i++) {
                    System.out.println(allItems.get(i));
                }

                if (allItems.size() > sampleSize) {
                    System.out.println("... and " + (allItems.size() - sampleSize) +
" more items");
                }
            }
        } catch (IllegalArgumentException e) {
            System.err.println("Invalid input: " + e.getMessage());
            System.exit(1);
        } catch (ResourceNotFoundException e) {
            System.err.println("Table not found: " + tableName);
            System.exit(1);
        } catch (DynamoDbException e) {
            System.err.println("DynamoDB error: " + e.getMessage());
            System.exit(1);
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
            System.exit(1);
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 Query。

通过强一致性读取查询表

以下代码示例演示如何使用强一致性读取来查询表。

- 为 DynamoDB 查询配置一致性级别。
- 使用强一致性读取来获取最多的 up-to-date数据。
- 了解最终一致性和强一致性之间的权衡。

适用于 Java 的 SDK 2.x

使用查询具有可配置读取一致性的 DynamoDB 表。 AWS SDK for Java 2.x

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithConsistentReads(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final boolean useConsistentRead) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);
```

```
        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .consistentRead(useConsistentRead)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query successful. Found {0} items",
response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying with consistent reads", e);
            throw e;
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 Query。

查询 TTL 项目

以下代码示例演示如何查询 TTL 项目。

适用于 Java 的 SDK 2.x

使用查询筛选表达式以收集 DynamoDB 表中的 TTL 项目。 AWS SDK for Java 2.x

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.util.Map;
import java.util.Optional;

        final QueryRequest request = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .filterExpression(FILTER_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try (DynamoDbClient ddb = dynamoDbClient != null
            ? dynamoDbClient
            : DynamoDbClient.builder().region(region).build()) {
            final QueryResponse response = ddb.query(request);
            System.out.println("Query successful. Found " + response.count() + "
 items that have not expired yet.");

            // Print each item
            response.items().forEach(item -> {
                System.out.println("Item: " + item);
            });

            return 0;
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 Query。

使用日期和时间模式查询表

以下代码示例演示如何使用日期和时间模式来查询表。

- 在 DynamoDB 中存储和查询 date/time 值。
- 使用排序键实现日期范围查询。
- 对日期字符串格式化以进行有效查询。

适用于 Java 的 SDK 2.x

使用排序键中的日期范围进行查询 AWS SDK for Java 2.x。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.time.LocalDate;
import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;

    public QueryResponse queryWithDateRange(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final LocalDate startDate,
        final LocalDate endDate) {

        // Focus on query logic, assuming parameters are valid
        if (startDate == null || endDate == null) {
            throw new IllegalArgumentException("Start date and end date cannot be
 null");
        }

        if (endDate.isBefore(startDate)) {
            throw new IllegalArgumentException("End date must be after start date");
```

```
        }

        // Format dates as ISO strings for DynamoDB (using just the date part)
        final String formattedStartDate = startDate.toString();
        final String formattedEndDate = endDate.toString();

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
partitionKeyName);
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_SK, dateKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_START_DATE,
            AttributeValue.builder().s(formattedStartDate).build());
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_END_DATE,
            AttributeValue.builder().s(formattedEndDate).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            LOGGER.log(Level.INFO, "Query by date range successful. Found {0}
items", response.count());
            return response;
        } catch (ResourceNotFoundException e) {
            LOGGER.log(Level.SEVERE, "Table not found: {0}", tableName);
            throw e;
        } catch (DynamoDbException e) {
            LOGGER.log(Level.SEVERE, "Error querying by date range: {0}",
e.getMessage());
```

```
            throw e;
        }
    }
```

使用带有的日期时间变量进行查询。 AWS SDK for Java 2.x

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.time.Instant;
import java.time.LocalDateTime;
import java.time.ZoneOffset;
import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryWithDateTime(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final String startDate,
        final String endDate) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateDateRangeParameters(dateKeyName, startDate,
 endDate);
        CodeSampleUtils.validateDateFormat("Start date", startDate);
        CodeSampleUtils.validateDateFormat("End date", endDate);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);
        expressionAttributeNames.put("#dateKey", dateKeyName);

        // Create expression attribute values for the column values
```

```
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            ":startDate", AttributeValue.builder().s(startDate).build());
        expressionAttributeValues.put(
            ":endDate", AttributeValue.builder().s(endDate).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query successful. Found " + response.count() + "
 items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
 found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with date range: " + e.getMessage());
            throw e;
        }
    }
```

使用 AWS SDK for Java 2.x在采用 Unix 纪元时间戳的日期范围内进行查询。

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;
```

```
import java.time.Instant;
import java.time.LocalDateTime;
import java.time.ZoneOffset;
import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryWithDateTimeEpoch(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final long startEpoch,
        final long endEpoch) {

        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
 partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Date key name", dateKeyName);
        CodeSampleUtils.validateEpochTimestamp("Start epoch", startEpoch);
        CodeSampleUtils.validateEpochTimestamp("End epoch", endEpoch);

        // Create expression attribute names for the column names
        final Map<String, String> expressionAttributeNames = new HashMap<>();
        expressionAttributeNames.put(EXPRESSION_ATTRIBUTE_NAME_PK,
 partitionKeyName);
        expressionAttributeNames.put("#dateKey", dateKeyName);

        // Create expression attribute values for the column values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            EXPRESSION_ATTRIBUTE_VALUE_PK,
            AttributeValue.builder().s(partitionKeyValue).build());
        expressionAttributeValues.put(
            ":startDate",
 AttributeValue.builder().n(String.valueOf(startEpoch)).build());
        expressionAttributeValues.put(
            ":endDate",
 AttributeValue.builder().n(String.valueOf(endEpoch)).build());

        // Create the query request
        final QueryRequest queryRequest = QueryRequest.builder()
            .tableName(tableName)
            .keyConditionExpression(KEY_CONDITION_EXPRESSION)
            .expressionAttributeNames(expressionAttributeNames)
```

```
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        try {
            final QueryResponse response = dynamoDbClient.query(queryRequest);
            System.out.println("Query successful. Found " + response.count() + "
items");
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format("Error: The Amazon DynamoDB table \"%s\" can't be
found.\n", tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println("Error querying with epoch timestamps: " +
e.getMessage());
            throw e;
        }
    }
```

使用带有的 LocalDateTime 对象在日期范围内进行查询 AWS SDK for Java 2.x。

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ResourceNotFoundException;

import java.time.Instant;
import java.time.LocalDateTime;
import java.time.ZoneOffset;
import java.util.HashMap;
import java.util.Map;

    public QueryResponse queryWithDateTimeLocalDateTime(
        final String tableName,
        final String partitionKeyName,
        final String partitionKeyValue,
        final String dateKeyName,
        final LocalDateTime startDateTime,
        final LocalDateTime endDateTime) {
```

```
        CodeSampleUtils.validateTableParameters(tableName, partitionKeyName,
partitionKeyValue);
        CodeSampleUtils.validateStringParameter("Date key name", dateKeyName);
        if (startDateTime == null || endDateTime == null) {
            throw new IllegalArgumentException("Start and end LocalDateTime must not
be null");
        }

        // Convert LocalDateTime to ISO-8601 strings in UTC with the correct format
        final String startDate =
startDateTime.atZone(ZoneOffset.UTC).format(DATE_TIME_FORMATTER);
        final String endDate =
endDateTime.atZone(ZoneOffset.UTC).format(DATE_TIME_FORMATTER);

        return queryWithDateTime(tableName, partitionKeyName, partitionKeyValue,
dateKeyName, startDate, endDate);
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 Query。

了解更新表达式顺序

以下代码示例演示如何了解更新表达式顺序。

- 了解 DynamoDB 如何处理更新表达式。
- 了解更新表达式中的操作顺序。
- 通过了解表达式求值来避免意外结果。

适用于 Java 的 SDK 2.x

使用演示更新表达式顺序 AWS SDK for Java 2.x。

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ReturnValue;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;
```

```
import java.util.HashMap;
import java.util.Map;

    /**
     * Demonstrates the effect of update expression order.
     *
     * <p>This method shows how the order of operations in an update expression
     * affects the result of the update.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @return Map containing the results of different update orders
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Map<String, Object> demonstrateUpdateOrder(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
 key) {

        Map<String, Object> results = new HashMap<>();

        try {
            // Initialize the item with a counter
            UpdateItemRequest initRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET Counter = :zero, OldCounter = :zero")
                .expressionAttributeValues(
                    Map.of(":zero", AttributeValue.builder().n("0").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            dynamoDbClient.updateItem(initRequest);

            // Example 1: SET first, then ADD
            UpdateItemRequest setFirstRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET Counter = :value ADD OldCounter :increment")
                .expressionAttributeValues(Map.of(
                    ":value", AttributeValue.builder().n("10").build(),
                    ":increment", AttributeValue.builder().n("5").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
```

```
                .build();

            UpdateItemResponse setFirstResponse =
dynamoDbClient.updateItem(setFirstRequest);
            results.put("setFirstResponse", setFirstResponse);

            // Reset the item
            dynamoDbClient.updateItem(initRequest);

            // Example 2: ADD first, then SET
            UpdateItemRequest addFirstRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("ADD Counter :increment SET OldCounter = :value")
                .expressionAttributeValues(Map.of(
                    ":value", AttributeValue.builder().n("10").build(),
                    ":increment", AttributeValue.builder().n("5").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse addFirstResponse =
dynamoDbClient.updateItem(addFirstRequest);
            results.put("addFirstResponse", addFirstResponse);

            // Reset the item
            dynamoDbClient.updateItem(initRequest);

            // Example 3: SET with multiple attributes
            UpdateItemRequest multiSetRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET Counter = :value, OldCounter = Counter")
                .expressionAttributeValues(
                    Map.of(":value", AttributeValue.builder().n("10").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse multiSetResponse =
dynamoDbClient.updateItem(multiSetRequest);
            results.put("multiSetResponse", multiSetResponse);

            // Reset the item
            dynamoDbClient.updateItem(initRequest);
```

```
            // Example 4: SET with expression using the same attribute
            UpdateItemRequest selfReferenceRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET Counter = Counter + :increment, OldCounter =
Counter")
                .expressionAttributeValues(
                    Map.of(":increment", AttributeValue.builder().n("5").build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse selfReferenceResponse =
dynamoDbClient.updateItem(selfReferenceRequest);
            results.put("selfReferenceResponse", selfReferenceResponse);

            results.put("success", true);

        } catch (DynamoDbException e) {
            results.put("success", false);
            results.put("error", e.getMessage());
        }

        return results;
    }

    /**
     * Updates an item with SET first, then REMOVE.
     *
     * <p>This method demonstrates updating an item with SET operation first,
     * followed by a REMOVE operation.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param attributeToSet The attribute to set
     * @param setValue The value to set
     * @param attributeToRemove The attribute to remove
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateWithSetFirst(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
```

```
        String attributeToSet,
        AttributeValue setValue,
        String attributeToRemove) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #setAttr = :setValue REMOVE #removeAttr")
            .expressionAttributeNames(Map.of(
                "#setAttr", attributeToSet,
                "#removeAttr", attributeToRemove))
            .expressionAttributeValues(Map.of(":setValue", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        try {
            return dynamoDbClient.updateItem(request);
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to update item with SET first: " + e.getMessage())
                .cause(e)
                .build();
        }
    }

    /**
     * Updates an item with REMOVE first, then SET.
     *
     * <p>This method demonstrates updating an item with REMOVE operation first,
     * followed by a SET operation.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param attributeToSet The attribute to set
     * @param setValue The value to set
     * @param attributeToRemove The attribute to remove
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateWithRemoveFirst(
        DynamoDbClient dynamoDbClient,
```

```
        String tableName,
        Map<String, AttributeValue> key,
        String attributeToSet,
        AttributeValue setValue,
        String attributeToRemove) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("REMOVE #removeAttr SET #setAttr = :setValue")
            .expressionAttributeNames(Map.of(
                "#setAttr", attributeToSet,
                "#removeAttr", attributeToRemove))
            .expressionAttributeValues(Map.of(":setValue", setValue))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        try {
            return dynamoDbClient.updateItem(request);
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to update item with REMOVE first: " +
e.getMessage())
                .cause(e)
                .build();
        }
    }

    /**
     * Updates an item with all operation types in a specific order.
     *
     * <p>This method demonstrates using all operation types (SET, REMOVE, ADD,
DELETE)
     * in a specific order in a single update expression.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateWithAllOperationTypes(
```

```
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #stringAttr = :stringVal, #mapAttr.#nestedAttr
= :nestedVal " + "REMOVE #oldAttr "
                + "ADD #counterAttr :increment "
                + "DELETE #stringSetAttr :stringSetVal")
            .expressionAttributeNames(Map.of(
                "#stringAttr", "StringAttribute",
                "#mapAttr", "MapAttribute",
                "#nestedAttr", "NestedAttribute",
                "#oldAttr", "OldAttribute",
                "#counterAttr", "CounterAttribute",
                "#stringSetAttr", "StringSetAttribute"))
            .expressionAttributeValues(Map.of(
                ":stringVal", AttributeValue.builder().s("New Value").build(),
                ":nestedVal", AttributeValue.builder().s("Nested Value").build(),
                ":increment", AttributeValue.builder().n("1").build(),
                ":stringSetVal", AttributeValue.builder().ss("Value1").build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        try {
            return dynamoDbClient.updateItem(request);
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to update item with all operation types: " +
e.getMessage())
                .cause(e)
                .build();
        }
    }

    /**
     * Gets the current state of an item.
     *
     * <p>Helper method to retrieve the current state of an item.
     *
     * @param dynamoDbClient The DynamoDB client
```

```
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @return The item or null if not found
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Map<String, AttributeValue> getItem(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key) {

        // Define the get parameters
        GetItemRequest request =
            GetItemRequest.builder().tableName(tableName).key(key).build();

        // Perform the get operation
        try {
            GetItemResponse response = dynamoDbClient.getItem(request);

            // Return the item if it exists, otherwise null
            return response.item();
        } catch (DynamoDbException e) {
            throw DynamoDbException.builder()
                .message("Failed to get item: " + e.getMessage())
                .cause(e)
                .build();
        }
    }
```

使用更新表达式顺序的用法示例 AWS SDK for Java 2.x。

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating update expression order in DynamoDB");

        try {
            // Example 1: Demonstrate update order effects
            System.out.println("\nExample 1: Demonstrating update order effects");
            Map<String, Object> orderResults =
demonstrateUpdateOrder(dynamoDbClient, tableName, key);
```

```
            if ((boolean) orderResults.get("success")) {
                System.out.println("SET first, then ADD:");
                System.out.println("  " + orderResults.get("setFirstResponse"));

                System.out.println("ADD first, then SET:");
                System.out.println("  " + orderResults.get("addFirstResponse"));

                System.out.println("SET with multiple attributes:");
                System.out.println("  " + orderResults.get("multiSetResponse"));

                System.out.println("SET with self-reference:");
                System.out.println("  " +
orderResults.get("selfReferenceResponse"));
            } else {
                System.out.println("Error: " + orderResults.get("error"));
            }

            // Example 2: Update with SET first, then REMOVE
            System.out.println("\nExample 2: Update with SET first, then REMOVE");
            UpdateItemResponse setFirstResponse = updateWithSetFirst(
                dynamoDbClient,
                tableName,
                key,
                "Status",
                AttributeValue.builder().s("Active").build(),
                "OldStatus");

            System.out.println("Updated attributes: " +
setFirstResponse.attributes());

            // Example 3: Update with REMOVE first, then SET
            System.out.println("\nExample 3: Update with REMOVE first, then SET");
            UpdateItemResponse removeFirstResponse = updateWithRemoveFirst(
                dynamoDbClient,
                tableName,
                key,
                "Status",
                AttributeValue.builder().s("Inactive").build(),
                "OldStatus");

            System.out.println("Updated attributes: " +
removeFirstResponse.attributes());
```

```
            // Example 4: Update with all operation types
            System.out.println("\nExample 4: Update with all operation types");
            UpdateItemResponse allOpsResponse =
updateWithAllOperationTypes(dynamoDbClient, tableName, key);

            System.out.println("Updated attributes: " +
allOpsResponse.attributes());

            // Example 5: Get the current state of the item
            System.out.println("\nExample 5: Current state of the item");
            Map<String, AttributeValue> item = getItem(dynamoDbClient, tableName,
key);

            if (item != null) {
                System.out.println("Item: " + item);
            } else {
                System.out.println("Item not found");
            }

            // Explain update expression order
            System.out.println("\nKey points about update expression order in
DynamoDB:");
            System.out.println("1. Update expressions are processed in this order:
SET, REMOVE, ADD, DELETE");
            System.out.println("2. Within each clause, operations are processed from
left to right");
            System.out.println("3. SET operations use the item state before any
updates in the expression");
            System.out.println("4. When an attribute is referenced multiple times,
the first operation wins");
            System.out.println("5. To reference a new value, split the update into
multiple operations");
            System.out.println("6. The order of clauses in the expression doesn't
change the evaluation order");
            System.out.println("7. For complex updates, consider using multiple
separate update operations");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考UpdateItem中的。

更新表的热吞吐量设置

以下代码示例演示如何更新表的热吞吐量设置。

适用于 Java 的 SDK 2.x

使用 AWS SDK for Java 2.x更新现有 DynamoDB 表上的热吞吐量设置。

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GlobalSecondaryIndexUpdate;
import
 software.amazon.awssdk.services.dynamodb.model.UpdateGlobalSecondaryIndexAction;
import software.amazon.awssdk.services.dynamodb.model.UpdateTableRequest;
import software.amazon.awssdk.services.dynamodb.model.WarmThroughput;

    public static WarmThroughput buildWarmThroughput(final Long readUnitsPerSecond,
 final Long writeUnitsPerSecond) {
        return WarmThroughput.builder()
            .readUnitsPerSecond(readUnitsPerSecond)
            .writeUnitsPerSecond(writeUnitsPerSecond)
            .build();
    }

    /**
     * Updates a DynamoDB table with warm throughput settings for both the table and
 a global secondary index.
     *
     * @param ddb The DynamoDB client
     * @param tableName The name of the table to update
     * @param tableReadUnitsPerSecond Read units per second for the table
     * @param tableWriteUnitsPerSecond Write units per second for the table
     * @param globalSecondaryIndexName The name of the global secondary index to
 update
     * @param globalSecondaryIndexReadUnitsPerSecond Read units per second for the
 GSI
     * @param globalSecondaryIndexWriteUnitsPerSecond Write units per second for the
 GSI
     */
    public static void updateDynamoDBTable(
        final DynamoDbClient ddb,
```

```
        final String tableName,
        final Long tableReadUnitsPerSecond,
        final Long tableWriteUnitsPerSecond,
        final String globalSecondaryIndexName,
        final Long globalSecondaryIndexReadUnitsPerSecond,
        final Long globalSecondaryIndexWriteUnitsPerSecond) {

        final WarmThroughput tableWarmThroughput =
            buildWarmThroughput(tableReadUnitsPerSecond, tableWriteUnitsPerSecond);
        final WarmThroughput gsiWarmThroughput =
            buildWarmThroughput(globalSecondaryIndexReadUnitsPerSecond,
globalSecondaryIndexWriteUnitsPerSecond);

        final GlobalSecondaryIndexUpdate globalSecondaryIndexUpdate =
GlobalSecondaryIndexUpdate.builder()
            .update(UpdateGlobalSecondaryIndexAction.builder()
                .indexName(globalSecondaryIndexName)
                .warmThroughput(gsiWarmThroughput)
                .build())
            .build();

        final UpdateTableRequest request = UpdateTableRequest.builder()
            .tableName(tableName)
            .globalSecondaryIndexUpdates(globalSecondaryIndexUpdate)
            .warmThroughput(tableWarmThroughput)
            .build();

        try {
            ddb.updateTable(request);
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }

        System.out.println(SUCCESS_MESSAGE);
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[UpdateTable](#)中的。

更新项目的 TTL

以下代码示例演示如何更新项目的 TTL。

适用于 Java 的 SDK 2.x

更新表中现有 DynamoDB 项目的 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 java.util.HashMap;
import java.util.Map;
import java.util.Optional;

    public UpdateItemResponse updateItemWithTTL(
        final String tableName, final String primaryKeyValue, final String
 sortKeyValue) {
        // Get current time in epoch second format
        final long currentTime = System.currentTimeMillis() / 1000;

        // Calculate expiration time 90 days from now in epoch second format
        final long expireDate = currentTime + (DAYS_TO_EXPIRE * SECONDS_PER_DAY);

        // Create the key map for the item to update
        final Map<String, AttributeValue> keyMap = new HashMap<>();
        keyMap.put(PRIMARY_KEY_ATTR,
 AttributeValue.builder().s(primaryKeyValue).build());
        keyMap.put(SORT_KEY_ATTR, AttributeValue.builder().s(sortKeyValue).build());

        // Create the expression attribute values
        final Map<String, AttributeValue> expressionAttributeValues = new
 HashMap<>();
        expressionAttributeValues.put(
            ":c", AttributeValue.builder().n(String.valueOf(currentTime)).build());
        expressionAttributeValues.put(
            ":e", AttributeValue.builder().n(String.valueOf(expireDate)).build());

        final UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(keyMap)
            .updateExpression(UPDATE_EXPRESSION)
            .expressionAttributeValues(expressionAttributeValues)
```

```
            .build();

        try {
            final UpdateItemResponse response = dynamoDbClient.updateItem(request);
            System.out.println(String.format(SUCCESS_MESSAGE, tableName));
            return response;
        } catch (ResourceNotFoundException e) {
            System.err.format(TABLE_NOT_FOUND_ERROR, tableName);
            throw e;
        } catch (DynamoDbException e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考UpdateItem中的。

使用 API Gateway 调用 Lambda 函数

以下代码示例展示了如何创建由 Amazon API Gateway 调用的 AWS Lambda 函数。

适用于 Java 的 SDK 2.x

演示如何使用 Lambda Java 运行时 API 创建 AWS Lambda 函数。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通过 Amazon API Gateway 调用的 Lambda 函数，该函数扫描 Amazon DynamoDB 表获取工作周年纪念日，并使用 Amazon Simple Notification Service (Amazon SNS)向员工发送文本消息，祝贺他们的周年纪念日。

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 Gateway
- DynamoDB
- Lambda
- Amazon SNS

使用 Step Functions 调用 Lambda 函数

以下代码示例说明如何创建按顺序调用 AWS Lambda 函数的 AWS Step Functions 状态机。

适用于 Java 的 SDK 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
- Amazon SES
- Step Functions

使用原子计数器操作

以下代码示例演示如何在 DynamoDB 中使用原子计数器操作。

- 使用 ADD 和 SET 操作以原子方式递增计数器。
- 安全地递增可能不存在的计数器。
- 为计数器操作实施乐观锁。

适用于 Java 的 SDK 2.x

使用演示原子计数器操作 AWS SDK for Java 2.x。

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ReturnValue;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.Map;

    /**
     * Increments a counter using the ADD operation.
     *
```

```
     * <p>This method demonstrates how to use the ADD operation to atomically
     * increment a counter attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param counterName The name of the counter attribute
     * @param incrementValue The value to increment by
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse incrementCounterWithAdd(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String counterName,
        int incrementValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("ADD #counterName :increment")
            .expressionAttributeNames(Map.of("#counterName", counterName))
            .expressionAttributeValues(Map.of(
                ":increment",
                AttributeValue.builder().n(String.valueOf(incrementValue)).build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Increments a counter using the SET operation.
     *
     * <p>This method demonstrates how to use the SET operation with an expression
     * to increment a counter attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param counterName The name of the counter attribute
```

```
     * @param incrementValue The value to increment by
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse incrementCounterWithSet(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String counterName,
        int incrementValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #counterName = #counterName + :increment")
            .expressionAttributeNames(Map.of("#counterName", counterName))
            .expressionAttributeValues(Map.of(
                ":increment",
                AttributeValue.builder().n(String.valueOf(incrementValue)).build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Increments a counter safely, handling the case where the counter doesn't
exist yet.
     *
     * <p>This method demonstrates how to use if_not_exists to safely increment a
counter
     * that may not exist yet.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param counterName The name of the counter attribute
     * @param incrementValue The value to increment by
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse incrementCounterSafely(
```

```
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String counterName,
        int incrementValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #counterName = if_not_exists(#counterName, :zero)
+ :increment")
            .expressionAttributeNames(Map.of("#counterName", counterName))
            .expressionAttributeValues(Map.of(
                ":increment",

AttributeValue.builder().n(String.valueOf(incrementValue)).build(),
                ":zero", AttributeValue.builder().n("0").build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Decrements a counter safely, ensuring it doesn't go below zero.
     *
     * <p>This method demonstrates how to use a condition expression to safely
     * decrement a counter without going below zero.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param counterName The name of the counter attribute
     * @param decrementValue The value to decrement by
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation or if the
counter would go below zero
     */
    public static UpdateItemResponse decrementCounterSafely(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
```

```
        String counterName,
        int decrementValue) {

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #counterName = #counterName - :decrement")
            .conditionExpression("#counterName >= :decrement")
            .expressionAttributeNames(Map.of("#counterName", counterName))
            .expressionAttributeValues(Map.of(
                ":decrement",
                AttributeValue.builder().n(String.valueOf(decrementValue)).build()))
            .returnValues(ReturnValue.UPDATED_NEW)
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Compares the ADD and SET approaches for incrementing counters.
     *
     * <p>This method demonstrates the differences between using ADD and SET
     * for incrementing counters in DynamoDB.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @return Map containing the comparison results
     */
    public static Map<String, Object> compareAddVsSet(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key) {

        Map<String, Object> results = new HashMap<>();

        try {
            // Reset counters to ensure a fair comparison
            UpdateItemRequest resetRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET AddCounter = :zero, SetCounter = :zero")
                .expressionAttributeValues(
```

```
                    Map.of(":zero", AttributeValue.builder().n("0").build()))
                .build();

            dynamoDbClient.updateItem(resetRequest);

            // Increment with ADD
            long addStartTime = System.nanoTime();
            UpdateItemResponse addResponse = incrementCounterWithAdd(dynamoDbClient,
tableName, key, "AddCounter", 1);
            long addEndTime = System.nanoTime();
            long addDuration = addEndTime - addStartTime;

            // Increment with SET
            long setStartTime = System.nanoTime();
            UpdateItemResponse setResponse = incrementCounterWithSet(dynamoDbClient,
tableName, key, "SetCounter", 1);
            long setEndTime = System.nanoTime();
            long setDuration = setEndTime - setStartTime;

            // Record results
            results.put("addResponse", addResponse);
            results.put("setResponse", setResponse);
            results.put("addDuration", addDuration);
            results.put("setDuration", setDuration);
            results.put("success", true);

        } catch (DynamoDbException e) {
            results.put("success", false);
            results.put("error", e.getMessage());
        }

        return results;
    }

    /**
     * Gets the current value of a counter attribute.
     *
     * <p>Helper method to retrieve the current value of a counter attribute.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to get
     * @param counterName The name of the counter attribute
     * @return The counter value or null if not found
```

```
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static Integer getCounterValue(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, String counterName) {

        // Define the get parameters
        GetItemRequest request = GetItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .projectionExpression(counterName)
            .build();

        // Perform the get operation
        GetItemResponse response = dynamoDbClient.getItem(request);

        // Return the counter value if it exists, otherwise null
        if (response.item() != null && response.item().containsKey(counterName)) {
            return Integer.parseInt(response.item().get(counterName).n());
        }

        return null;
    }
```

使用原子计数器操作的用法示例 AWS SDK for Java 2.x。

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating atomic counter operations in DynamoDB");

        try {
            // Example 1: Increment a counter using ADD
            System.out.println("\nExample 1: Incrementing a counter using ADD");
            UpdateItemResponse addResponse = incrementCounterWithAdd(dynamoDbClient,
tableName, key, "ViewCount", 1);

            System.out.println("Updated counter: " + addResponse.attributes());
```

```
            // Example 2: Increment a counter using SET
            System.out.println("\nExample 2: Incrementing a counter using SET");
            UpdateItemResponse setResponse = incrementCounterWithSet(dynamoDbClient,
tableName, key, "LikeCount", 1);

            System.out.println("Updated counter: " + setResponse.attributes());

            // Example 3: Increment a counter safely
            System.out.println("\nExample 3: Incrementing a counter safely");
            UpdateItemResponse safeResponse = incrementCounterSafely(dynamoDbClient,
tableName, key, "ShareCount", 1);

            System.out.println("Updated counter: " + safeResponse.attributes());

            // Example 4: Decrement a counter safely
            System.out.println("\nExample 4: Decrementing a counter safely");
            try {
                UpdateItemResponse decrementResponse =
                    decrementCounterSafely(dynamoDbClient, tableName, key,
"InventoryCount", 1);

                System.out.println("Updated counter: " +
decrementResponse.attributes());
            } catch (DynamoDbException e) {
                if (e.getMessage().contains("ConditionalCheckFailed")) {
                    System.out.println("Cannot decrement counter below zero");
                } else {
                    throw e;
                }
            }

            // Example 5: Compare ADD vs SET
            System.out.println("\nExample 5: Comparing ADD vs SET");
            Map<String, Object> comparison = compareAddVsSet(dynamoDbClient,
tableName, key);

            if ((boolean) comparison.get("success")) {
                System.out.println("ADD duration: " + comparison.get("addDuration")
+ " ns");
                System.out.println("SET duration: " + comparison.get("setDuration")
+ " ns");
                System.out.println("ADD response: " +
comparison.get("addResponse"));
```

```
                System.out.println("SET response: " +
comparison.get("setResponse"));
            } else {
                System.out.println("Comparison failed: " + comparison.get("error"));
            }

            // Explain atomic counter operations
            System.out.println("\nKey points about DynamoDB atomic counter
operations:");
            System.out.println("1. Both ADD and SET can be used for atomic
counters");
            System.out.println("2. ADD is more concise for simple increments");
            System.out.println("3. SET with an expression is more flexible for
complex operations");
            System.out.println("4. Use if_not_exists to handle the case where the
counter doesn't exist yet");
            System.out.println("5. Use condition expressions to prevent counters
from going below zero");
            System.out.println("6. Atomic operations are guaranteed to be isolated
from other writes");
            System.out.println("7. ADD can only be used with number and set data
types");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[UpdateItem](#)中的。

使用条件运算

以下代码示例演示如何在 DynamoDB 中使用条件运算。

- 实现条件写入以防止覆盖数据。
- 使用条件表达式以强制实施业务规则。
- 从容处理条件检查失败。

适用于 Java 的 SDK 2.x

使用演示条件运算 AWS SDK for Java 2.x。

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import
 software.amazon.awssdk.services.dynamodb.model.ConditionalCheckFailedException;
import software.amazon.awssdk.services.dynamodb.model.DeleteItemRequest;
import software.amazon.awssdk.services.dynamodb.model.DeleteItemResponse;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.GetItemRequest;
import software.amazon.awssdk.services.dynamodb.model.GetItemResponse;
import software.amazon.awssdk.services.dynamodb.model.ReturnValue;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.HashMap;
import java.util.Map;

    /**
     * Performs a conditional update on an item.
     *
     * <p>This method demonstrates how to use a condition expression to update an
 item
     * only if a specific condition is met.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param conditionAttribute The attribute to check in the condition
     * @param conditionValue The value to compare against
     * @param updateAttribute The attribute to update
     * @param updateValue The new value to set
     * @return Map containing the operation result and status
     */
    public static Map<String, Object> conditionalUpdate(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String conditionAttribute,
        AttributeValue conditionValue,
        String updateAttribute,
        AttributeValue updateValue) {
```

```
        Map<String, Object> result = new HashMap<>();

        try {
            // Define the update parameters
            UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #updateAttr = :updateVal")
                .conditionExpression("#condAttr = :condVal")
                .expressionAttributeNames(Map.of(
                    "#condAttr", conditionAttribute,
                    "#updateAttr", updateAttribute))
                .expressionAttributeValues(Map.of(
                    ":condVal", conditionValue,
                    ":updateVal", updateValue))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            // Perform the update operation
            UpdateItemResponse response = dynamoDbClient.updateItem(request);

            // Record success result
            result.put("success", true);
            result.put("message", "Condition was met and update was performed");
            result.put("attributes", response.attributes());

        } catch (ConditionalCheckFailedException e) {
            // Record failure due to condition not being met
            result.put("success", false);
            result.put("message", "Condition was not met, update was not
performed");
            result.put("error", "ConditionalCheckFailedException");

        } catch (DynamoDbException e) {
            // Record failure due to other errors
            result.put("success", false);
            result.put("message", "Error occurred: " + e.getMessage());
            result.put("error", e.getClass().getSimpleName());
        }

        return result;
    }
```

```
    /**
     * Performs a conditional delete on an item.
     *
     * <p>This method demonstrates how to use a condition expression to delete an
item
     * only if a specific condition is met.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to delete
     * @param conditionAttribute The attribute to check in the condition
     * @param conditionValue The value to compare against
     * @return Map containing the operation result and status
     */
    public static Map<String, Object> conditionalDelete(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String conditionAttribute,
        AttributeValue conditionValue) {

        Map<String, Object> result = new HashMap<>();

        try {
            // Define the delete parameters
            DeleteItemRequest request = DeleteItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .conditionExpression("#condAttr = :condVal")
                .expressionAttributeNames(Map.of("#condAttr", conditionAttribute))
                .expressionAttributeValues(Map.of(":condVal", conditionValue))
                .returnValues(ReturnValue.ALL_OLD)
                .build();

            // Perform the delete operation
            DeleteItemResponse response = dynamoDbClient.deleteItem(request);

            // Record success result
            result.put("success", true);
            result.put("message", "Condition was met and delete was performed");
            result.put("attributes", response.attributes());

        } catch (ConditionalCheckFailedException e) {
            // Record failure due to condition not being met
```

```
            result.put("success", false);
            result.put("message", "Condition was not met, delete was not
performed");
            result.put("error", "ConditionalCheckFailedException");

        } catch (DynamoDbException e) {
            // Record failure due to other errors
            result.put("success", false);
            result.put("message", "Error occurred: " + e.getMessage());
            result.put("error", e.getClass().getSimpleName());
        }

        return result;
    }

    /**
     * Demonstrates optimistic locking using a version attribute.
     *
     * <p>This method shows how to implement optimistic locking by using a version
     * attribute that is incremented with each update.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param versionAttribute The name of the version attribute
     * @return Map containing the operation result
     */
    public static Map<String, Object> optimisticLockingExample(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
key, String versionAttribute) {

        Map<String, Object> result = new HashMap<>();

        try {
            // Get the current version of the item
            GetItemRequest getRequest = GetItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .projectionExpression(versionAttribute)
                .build();

            GetItemResponse getResponse = dynamoDbClient.getItem(getRequest);

            // Check if the item exists
```

```
            if (getResponse.item() == null || !
getResponse.item().containsKey(versionAttribute)) {
                // Item doesn't exist or doesn't have a version attribute
                // Initialize with version 1
                UpdateItemRequest initRequest = UpdateItemRequest.builder()
                    .tableName(tableName)
                    .key(key)
                    .updateExpression("SET #verAttr = :newVer, #dataAttr = :data")
                    .expressionAttributeNames(Map.of("#verAttr", versionAttribute,
 "#dataAttr", "Data"))
                    .expressionAttributeValues(Map.of(
                        ":newVer", AttributeValue.builder().n("1").build(),
                        ":data", AttributeValue.builder().s("Initial
 data").build()))
                    .returnValues(ReturnValue.UPDATED_NEW)
                    .build();

                UpdateItemResponse initResponse =
 dynamoDbClient.updateItem(initRequest);

                result.put("operation", "initialize");
                result.put("success", true);
                result.put("attributes", initResponse.attributes());

                return result;
            }

            // Get the current version number
            int currentVersion =
                Integer.parseInt(getResponse.item().get(versionAttribute).n());
            int newVersion = currentVersion + 1;

            // Update the item with a condition on the version
            UpdateItemRequest updateRequest = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #verAttr = :newVer, #dataAttr = :newData")
                .conditionExpression("#verAttr = :curVer")
                .expressionAttributeNames(Map.of("#verAttr", versionAttribute,
 "#dataAttr", "Data"))
                .expressionAttributeValues(Map.of(
                    ":curVer",
                        AttributeValue.builder()
                            .n(String.valueOf(currentVersion))
```

```
                            .build(),
                    ":newVer",

AttributeValue.builder().n(String.valueOf(newVersion)).build(),
                    ":newData",
                        AttributeValue.builder()
                            .s("Updated data at version " + newVersion)
                            .build()))
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            UpdateItemResponse updateResponse =
dynamoDbClient.updateItem(updateRequest);

            // Record success result
            result.put("operation", "update");
            result.put("success", true);
            result.put("oldVersion", currentVersion);
            result.put("newVersion", newVersion);
            result.put("attributes", updateResponse.attributes());

        } catch (ConditionalCheckFailedException e) {
            // Record failure due to version mismatch
            result.put("operation", "update");
            result.put("success", false);
            result.put("message", "Version mismatch, another process may have
updated the item");
            result.put("error", "ConditionalCheckFailedException");

        } catch (DynamoDbException e) {
            // Record failure due to other errors
            result.put("operation", "update");
            result.put("success", false);
            result.put("message", "Error occurred: " + e.getMessage());
            result.put("error", e.getClass().getSimpleName());
        }

        return result;
    }

    /**
     * Performs a conditional update with multiple conditions.
     *
```

```
    * <p>This method demonstrates how to use multiple conditions in a condition
expression.
    *
    * @param dynamoDbClient The DynamoDB client
    * @param tableName The name of the DynamoDB table
    * @param key The key of the item to update
    * @param conditions Map of attribute names to values for conditions
    * @param updateAttribute The attribute to update
    * @param updateValue The new value to set
    * @return Map containing the operation result and status
    */
   public static Map<String, Object> conditionalUpdateWithMultipleConditions(
       DynamoDbClient dynamoDbClient,
       String tableName,
       Map<String, AttributeValue> key,
       Map<String, AttributeValue> conditions,
       String updateAttribute,
       AttributeValue updateValue) {

       Map<String, Object> result = new HashMap<>();

       try {
           // Build the condition expression and attribute names/values
           StringBuilder conditionExpression = new StringBuilder();
           Map<String, String> expressionAttributeNames = new HashMap<>();
           Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();

           // Add update attribute
           expressionAttributeNames.put("#updateAttr", updateAttribute);
           expressionAttributeValues.put(":updateVal", updateValue);

           // Add conditions
           int i = 0;
           for (Map.Entry<String, AttributeValue> condition :
conditions.entrySet()) {
               String attrName = condition.getKey();
               AttributeValue attrValue = condition.getValue();

               String nameKey = "#cond" + i;
               String valueKey = ":val" + i;

               expressionAttributeNames.put(nameKey, attrName);
               expressionAttributeValues.put(valueKey, attrValue);
```

```
                // Add AND between conditions (except for the first one)
                if (i > 0) {
                    conditionExpression.append(" AND ");
                }

                conditionExpression.append(nameKey).append(" = ").append(valueKey);
                i++;
            }

            // Define the update parameters
            UpdateItemRequest request = UpdateItemRequest.builder()
                .tableName(tableName)
                .key(key)
                .updateExpression("SET #updateAttr = :updateVal")
                .conditionExpression(conditionExpression.toString())
                .expressionAttributeNames(expressionAttributeNames)
                .expressionAttributeValues(expressionAttributeValues)
                .returnValues(ReturnValue.UPDATED_NEW)
                .build();

            // Perform the update operation
            UpdateItemResponse response = dynamoDbClient.updateItem(request);

            // Record success result
            result.put("success", true);
            result.put("message", "All conditions were met and update was
performed");
            result.put("attributes", response.attributes());

        } catch (ConditionalCheckFailedException e) {
            // Record failure due to condition not being met
            result.put("success", false);
            result.put("message", "One or more conditions were not met, update was
not performed");
            result.put("error", "ConditionalCheckFailedException");

        } catch (DynamoDbException e) {
            // Record failure due to other errors
            result.put("success", false);
            result.put("message", "Error occurred: " + e.getMessage());
            result.put("error", e.getClass().getSimpleName());
        }

        return result;
```

```
    }
```

条件运算的用法示例 AWS SDK for Java 2.x。

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating conditional operations in DynamoDB");

        try {
            // Example 1: Conditional update
            System.out.println("\nExample 1: Conditional update");
            Map<String, Object> updateResult = conditionalUpdate(
                dynamoDbClient,
                tableName,
                key,
                "InStock",
                AttributeValue.builder().bool(true).build(),
                "Status",
                AttributeValue.builder().s("Available").build());

            System.out.println("Update result: " + updateResult.get("message"));
            if ((boolean) updateResult.get("success")) {
                System.out.println("Updated attributes: " +
updateResult.get("attributes"));
            }

            // Example 2: Conditional delete
            System.out.println("\nExample 2: Conditional delete");
            Map<String, Object> deleteResult = conditionalDelete(
                dynamoDbClient,
                tableName,
                key,
                "Status",
                AttributeValue.builder().s("Discontinued").build());

            System.out.println("Delete result: " + deleteResult.get("message"));
            if ((boolean) deleteResult.get("success")) {
```

```
                System.out.println("Deleted item: " +
deleteResult.get("attributes"));
            }

            // Example 3: Optimistic locking
            System.out.println("\nExample 3: Optimistic locking");
            Map<String, Object> lockingResult =
optimisticLockingExample(dynamoDbClient, tableName, key, "Version");

            System.out.println("Optimistic locking result:");
            System.out.println("  Operation: " + lockingResult.get("operation"));
            System.out.println("  Success: " + lockingResult.get("success"));
            if (lockingResult.get("operation").equals("update") && (boolean)
lockingResult.get("success")) {
                System.out.println("  Old version: " +
lockingResult.get("oldVersion"));
                System.out.println("  New version: " +
lockingResult.get("newVersion"));
            }
            System.out.println("  Attributes: " + lockingResult.get("attributes"));

            // Example 4: Multiple conditions
            System.out.println("\nExample 4: Multiple conditions");
            Map<String, AttributeValue> conditions = new HashMap<>();
            conditions.put("Price", AttributeValue.builder().n("199.99").build());
            conditions.put("Category",
AttributeValue.builder().s("Electronics").build());

            Map<String, Object> multiConditionResult =
conditionalUpdateWithMultipleConditions(
                dynamoDbClient,
                tableName,
                key,
                conditions,
                "OnSale",
                AttributeValue.builder().bool(true).build());

            System.out.println("Multiple conditions result: " +
multiConditionResult.get("message"));
            if ((boolean) multiConditionResult.get("success")) {
                System.out.println("Updated attributes: " +
multiConditionResult.get("attributes"));
            }
```

```
            // Explain conditional operations
            System.out.println("\nKey points about DynamoDB conditional
operations:");
            System.out.println("1. Conditional operations only succeed if the
condition is met");
            System.out.println("2. ConditionalCheckFailedException is thrown when
the condition fails");
            System.out.println("3. No changes are made to the item if the condition
fails");
            System.out.println("4. Conditions can be used with update, delete, and
put operations");
            System.out.println("5. Multiple conditions can be combined with AND and
OR");
            System.out.println("6. Optimistic locking can be implemented using a
version attribute");
            System.out.println(
                "7. Conditional operations consume the same amount of write capacity
whether they succeed or fail");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - DeleteItem
  - PutItem
  - UpdateItem

使用表达式属性名称

以下代码示例演示如何在 DynamoDB 中使用表达式属性名称。

- 在 DynamoDB 表达式中使用保留字。
- 使用表达式属性名称占位符。
- 处理属性名称中的特殊字符。

适用于 Java 的 SDK 2.x

使用演示表达式属性名称 AWS SDK for Java 2.x。

```
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.AttributeValue;
import software.amazon.awssdk.services.dynamodb.model.DynamoDbException;
import software.amazon.awssdk.services.dynamodb.model.QueryRequest;
import software.amazon.awssdk.services.dynamodb.model.QueryResponse;
import software.amazon.awssdk.services.dynamodb.model.ScanRequest;
import software.amazon.awssdk.services.dynamodb.model.ScanResponse;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemRequest;
import software.amazon.awssdk.services.dynamodb.model.UpdateItemResponse;

import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

    /**
     * Updates an attribute that is a reserved word in DynamoDB.
     *
     * <p>This method demonstrates how to use expression attribute names to update
     * attributes that are reserved words in DynamoDB.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param reservedWordAttribute The reserved word attribute to update
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateReservedWordAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String reservedWordAttribute,
        AttributeValue value) {

        // Define the update parameters using expression attribute names
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
```

```
            .updateExpression("SET #attr = :value")
            .expressionAttributeNames(Map.of("#attr", reservedWordAttribute))
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Updates an attribute that contains special characters.
     *
     * <p>This method demonstrates how to use expression attribute names to update
     * attributes that contain special characters.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param specialCharAttribute The attribute with special characters to update
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateSpecialCharacterAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        String specialCharAttribute,
        AttributeValue value) {

        // Define the update parameters using expression attribute names
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET #attr = :value")
            .expressionAttributeNames(Map.of("#attr", specialCharAttribute))
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }
```

```
/**
 * Queries items using an attribute that is a reserved word.
 *
 * <p>This method demonstrates how to use expression attribute names in a query
 * when the attribute is a reserved word.
 *
 * @param dynamoDbClient The DynamoDB client
 * @param tableName The name of the DynamoDB table
 * @param partitionKeyName The name of the partition key attribute
 * @param partitionKeyValue The value of the partition key
 * @param reservedWordAttribute The reserved word attribute to filter on
 * @param value The value to compare against
 * @return The response from DynamoDB
 * @throws DynamoDbException if an error occurs during the operation
 */
public static QueryResponse queryWithReservedWordAttribute(
    DynamoDbClient dynamoDbClient,
    String tableName,
    String partitionKeyName,
    AttributeValue partitionKeyValue,
    String reservedWordAttribute,
    AttributeValue value) {

    // Define the query parameters using expression attribute names
    Map<String, String> expressionAttributeNames = new HashMap<>();
    expressionAttributeNames.put("#pkName", partitionKeyName);
    expressionAttributeNames.put("#attr", reservedWordAttribute);

    Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
    expressionAttributeValues.put(":pkValue", partitionKeyValue);
    expressionAttributeValues.put(":value", value);

    QueryRequest request = QueryRequest.builder()
        .tableName(tableName)
        .keyConditionExpression("#pkName = :pkValue")
        .filterExpression("#attr = :value")
        .expressionAttributeNames(expressionAttributeNames)
        .expressionAttributeValues(expressionAttributeValues)
        .build();

    // Perform the query operation
    return dynamoDbClient.query(request);
}
```

```
    /**
     * Updates a nested attribute with a path that contains reserved words.
     *
     * <p>This method demonstrates how to use expression attribute names to update
     * nested attributes where the path contains reserved words.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param key The key of the item to update
     * @param attributePath The path to the nested attribute as an array
     * @param value The value to set
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static UpdateItemResponse updateNestedReservedWordAttribute(
        DynamoDbClient dynamoDbClient,
        String tableName,
        Map<String, AttributeValue> key,
        List<String> attributePath,
        AttributeValue value) {

        // Create expression attribute names for each part of the path
        Map<String, String> expressionAttributeNames = new HashMap<>();
        for (int i = 0; i < attributePath.size(); i++) {
            expressionAttributeNames.put("#attr" + i, attributePath.get(i));
        }

        // Build the attribute path using the expression attribute names
        StringBuilder attributePathExpression = new StringBuilder();
        for (int i = 0; i < attributePath.size(); i++) {
            if (i > 0) {
                attributePathExpression.append(".");
            }
            attributePathExpression.append("#attr").append(i);
        }

        // Define the update parameters
        UpdateItemRequest request = UpdateItemRequest.builder()
            .tableName(tableName)
            .key(key)
            .updateExpression("SET " + attributePathExpression.toString() + "
= :value")
            .expressionAttributeNames(expressionAttributeNames)
```

```
            .expressionAttributeValues(Map.of(":value", value))
            .returnValues("UPDATED_NEW")
            .build();

        // Perform the update operation
        return dynamoDbClient.updateItem(request);
    }

    /**
     * Scans a table with multiple attribute name placeholders.
     *
     * <p>This method demonstrates how to use multiple expression attribute names
     * in a complex filter expression.
     *
     * @param dynamoDbClient The DynamoDB client
     * @param tableName The name of the DynamoDB table
     * @param filters Object mapping attribute names to filter values
     * @return The response from DynamoDB
     * @throws DynamoDbException if an error occurs during the operation
     */
    public static ScanResponse scanWithMultipleAttributeNames(
        DynamoDbClient dynamoDbClient, String tableName, Map<String, AttributeValue>
filters) {

        // Create expression attribute names and values
        Map<String, String> expressionAttributeNames = new HashMap<>();
        Map<String, AttributeValue> expressionAttributeValues = new HashMap<>();
        StringBuilder filterExpression = new StringBuilder();

        // Build the filter expression
        int index = 0;
        for (Map.Entry<String, AttributeValue> entry : filters.entrySet()) {
            String attrName = entry.getKey();
            AttributeValue attrValue = entry.getValue();

            String nameKey = "#attr" + index;
            String valueKey = ":val" + index;

            expressionAttributeNames.put(nameKey, attrName);
            expressionAttributeValues.put(valueKey, attrValue);

            // Add AND between conditions (except for the first one)
            if (index > 0) {
                filterExpression.append(" AND ");
```

```
            }

            filterExpression.append(nameKey).append(" = ").append(valueKey);
            index++;
        }

        // Define the scan parameters
        ScanRequest request = ScanRequest.builder()
            .tableName(tableName)
            .filterExpression(filterExpression.toString())
            .expressionAttributeNames(expressionAttributeNames)
            .expressionAttributeValues(expressionAttributeValues)
            .build();

        // Perform the scan operation
        return dynamoDbClient.scan(request);
    }
```

表达式属性名称的用法示例 AWS SDK for Java 2.x。

```
    public static void exampleUsage(DynamoDbClient dynamoDbClient, String tableName)
{
        // Example key
        Map<String, AttributeValue> key = new HashMap<>();
        key.put("ProductId", AttributeValue.builder().s("P12345").build());

        System.out.println("Demonstrating expression attribute names in DynamoDB");

        try {
            // Example 1: Update an attribute that is a reserved word
            System.out.println("\nExample 1: Updating an attribute that is a
reserved word");
            UpdateItemResponse response1 = updateReservedWordAttribute(
                dynamoDbClient,
                tableName,
                key,
                "Size", // "SIZE" is a reserved word in DynamoDB
                AttributeValue.builder().s("Large").build());

            System.out.println("Updated attribute: " + response1.attributes());

            // Example 2: Update an attribute with special characters
```

```
            System.out.println("\nExample 2: Updating an attribute with special
characters");
            UpdateItemResponse response2 = updateSpecialCharacterAttribute(
                dynamoDbClient,
                tableName,
                key,
                "Product-Type", // Contains a hyphen, which is a special character
                AttributeValue.builder().s("Electronics").build());

            System.out.println("Updated attribute: " + response2.attributes());

            // Example 3: Query with a reserved word attribute
            System.out.println("\nExample 3: Querying with a reserved word
attribute");
            QueryResponse response3 = queryWithReservedWordAttribute(
                dynamoDbClient,
                tableName,
                "Category",
                AttributeValue.builder().s("Electronics").build(),
                "Count", // "COUNT" is a reserved word in DynamoDB
                AttributeValue.builder().n("10").build());

            System.out.println("Found " + response3.count() + " items");

            // Example 4: Update a nested attribute with reserved words in the path
            System.out.println("\nExample 4: Updating a nested attribute with
reserved words in the path");
            UpdateItemResponse response4 = updateNestedReservedWordAttribute(
                dynamoDbClient,
                tableName,
                key,
                Arrays.asList("Dimensions", "Size", "Height"), // "SIZE" is a
reserved word
                AttributeValue.builder().n("30").build());

            System.out.println("Updated nested attribute: " +
response4.attributes());

            // Example 5: Scan with multiple attribute name placeholders
            System.out.println("\nExample 5: Scanning with multiple attribute name
placeholders");
            Map<String, AttributeValue> filters = new HashMap<>();
            filters.put("Size", AttributeValue.builder().s("Large").build());
            filters.put("Count", AttributeValue.builder().n("10").build());
```

```
            filters.put(
                "Product-Type", AttributeValue.builder().s("Electronics").build());

            ScanResponse response5 = scanWithMultipleAttributeNames(dynamoDbClient,
tableName, filters);

            System.out.println("Found " + response5.count() + " items");

            // Show some common reserved words
            System.out.println("\nSome common DynamoDB reserved words:");
            List<String> commonReservedWords = getDynamoDBReservedWords();
            System.out.println(String.join(", ", commonReservedWords));

            // Explain expression attribute names
            System.out.println("\nKey points about expression attribute names:");
            System.out.println("1. Use expression attribute names (#name) for
reserved words");
            System.out.println("2. Use expression attribute names for attributes
with special characters");
            System.out.println(
                "3. Special characters include: spaces, hyphens, dots, and other
non-alphanumeric characters");
            System.out.println("4. Expression attribute names are required for
nested attributes with reserved words");
            System.out.println("5. You can use multiple expression attribute names
in a single expression");
            System.out.println("6. Expression attribute names are case-sensitive");
            System.out.println("7. Expression attribute names are only used in
expressions, not in the actual data");

        } catch (DynamoDbException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - Query
  - UpdateItem

使用计划的事件调用 Lambda 函数

以下代码示例显示如何创建由 Amazon EventBridge 计划事件调用的 AWS Lambda 函数。

适用于 Java 的 SDK 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。在本示例中，您使用 Lambda Java 运行时 API 创建 Lambda 函数。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

- CloudWatch 日志
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

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一个 Lambda 函数，该函数接收通过接收来自 DynamoDB 流的记录而触发的事件。该函数检索 DynamoDB 有效负载，并记录下记录内容。

适用于 Java 的 SDK 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。

适用于 Java 的 SDK 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.util.ArrayList;
import java.util.List;

public class ProcessDynamodbRecords implements RequestHandler<DynamodbEvent,
 StreamsEventResponse> {

    @Override
    public StreamsEventResponse handleRequest(DynamodbEvent input, Context context)
 {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<>();
        String curRecordSequenceNumber = "";

        for (DynamodbEvent.DynamodbStreamRecord dynamodbStreamRecord :
 input.getRecords()) {
          try {
                //Process your record
                StreamRecord dynamodbRecord = dynamodbStreamRecord.getDynamodb();
                curRecordSequenceNumber = dynamodbRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed item
 immediately.
                   Lambda will immediately begin to retry processing from this
 failed item onwards. */
                batchItemFailures.add(new
 StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

       return new StreamsEventResponse();
    }
}
```

AWS 社区捐款

构建和测试无服务器应用程序

以下代码示例演示如何使用 API Gateway 与 Lambda 和 DynamoDB 构建和测试无服务器应用程序

适用于 Java 的 SDK 2.x

演示如何使用 Java SDK 构建和测试包含 API Gateway 以及 Lambda 和 DynamoDB 的无服务器应用程序。

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

- API Gateway
- DynamoDB
- Lambda

使用适用于 Java 的 SDK 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

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 基本功能
- 操作
- 场景

开始使用

你好 Amazon EC2

以下代码示例显示了如何开始使用 Amazon EC2。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Asynchronously describes the security groups for the specified group ID.
    *
    * @param groupName the name of the security group to describe
    * @return a {@link CompletableFuture} that represents the asynchronous
operation
    *         of describing the security groups. The future will complete with a
    *         {@link DescribeSecurityGroupsResponse} object that contains the
    *         security group information.
    */
   public CompletableFuture<String> describeSecurityGroupArnByNameAsync(String
groupName) {
       DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
           .groupNames(groupName)
           .build();

       DescribeSecurityGroupsPublisher paginator =
getAsyncClient().describeSecurityGroupsPaginator(request);
       AtomicReference<String> groupIdRef = new AtomicReference<>();
       return paginator.subscribe(response -> {
           response.securityGroups().stream()
               .filter(securityGroup ->
securityGroup.groupName().equals(groupName))
               .findFirst()
               .ifPresent(securityGroup ->
groupIdRef.set(securityGroup.groupId()));
```

```
        }).thenApply(v -> {
            String groupId = groupIdRef.get();
            if (groupId == null) {
                throw new RuntimeException("No security group found with the name: "
+ groupName);
            }
            return groupId;
        }).exceptionally(ex -> {
            logger.info("Failed to describe security group: " + ex.getMessage());
            throw new RuntimeException("Failed to describe security group", ex);
        });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeSecurityGroups中的。

基本功能

了解基本功能

以下代码示例展示了如何：

- 创建密钥对和安全组。
- 选择 Amazon 机器映像 (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 SSH 连接到您的实例，然后清理资源。

适用于 Java 的 SDK 2.x

> **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场景。

```
import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.ec2.model.CreateKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.DeleteKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.DescribeKeyPairsResponse;
import software.amazon.awssdk.services.ec2.model.DisassociateAddressResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.ReleaseAddressResponse;
import software.amazon.awssdk.services.ssm.model.GetParametersByPathResponse;
import software.amazon.awssdk.services.ssm.model.Parameter;

import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Creates an RSA key pair and saves the private key data as a .pem file.
 * 2. Lists key pairs.
 * 3. Creates a security group for the default VPC.
 * 4. Displays security group information.
 * 5. Gets a list of Amazon Linux 2 AMIs and selects one.
 * 6. Gets additional information about the image.
 * 7. Gets a list of instance types that are compatible with the selected AMI's
 * architecture.
 * 8. Creates an instance with the key pair, security group, AMI, and an
 * instance type.
 * 9. Displays information about the instance.
 * 10. Stops the instance and waits for it to stop.
 * 11. Starts the instance and waits for it to start.
 * 12. Allocates an Elastic IP address and associates it with the instance.
 * 13. Displays SSH connection info for the instance.
 * 14. Disassociates and deletes the Elastic IP address.
 * 15. Terminates the instance and waits for it to terminate.
```

```
 * 16. Deletes the security group.
 * 17. Deletes the key pair.
 */
public class EC2Scenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final Logger logger = LoggerFactory.getLogger(EC2Scenario.class);
    public static void main(String[] args) throws InterruptedException,
 UnknownHostException {

        logger.info("""
            Usage:
               <keyName> <fileName> <groupName> <groupDesc>

            Where:
               keyName -  A key pair name (for example, TestKeyPair).\s
               fileName -  A file name where the key information is written to.\s
               groupName - The name of the security group.\s
               groupDesc - The description of the security group.\s
            """);

        Scanner scanner = new Scanner(System.in);
        EC2Actions ec2Actions = new EC2Actions();

        String keyName = "TestKeyPair7" ;
        String fileName = "ec2Key.pem";
        String groupName = "TestSecGroup7" ;
        String groupDesc = "Test Group" ;
        String vpcId = ec2Actions.describeFirstEC2VpcAsync().join().vpcId();
        InetAddress localAddress = InetAddress.getLocalHost();
        String myIpAddress = localAddress.getHostAddress();

        logger.info("""
            Amazon Elastic Compute Cloud (EC2) is a web service that provides
 secure, resizable compute
            capacity in the cloud. It allows developers and organizations to easily
 launch and manage
            virtual server instances, known as EC2 instances, to run their
 applications.

            EC2 provides a wide range of instance types, each with different
 compute, memory,
            and storage capabilities, to meet the diverse needs of various
 workloads. Developers
```

```
            can choose the appropriate instance type based on their application's
 requirements,
            such as high-performance computing, memory-intensive tasks, or GPU-
accelerated workloads.

            The `Ec2AsyncClient` interface in the AWS SDK for Java 2.x provides a
 set of methods to
            programmatically interact with the Amazon EC2 service. This allows
 developers to
            automate the provisioning, management, and monitoring of EC2 instances
 as part of their
            application deployment pipelines. With EC2, teams can focus on building
 and deploying
            their applications without having to worry about the underlying
 infrastructure
            required to host and manage physical servers.

            This scenario walks you through how to perform key operations for this
 service.
            Let's get started...
            """);

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("1. Create an RSA key pair and save the private key material as
 a .pem file.");
        logger.info("""
            An RSA key pair for Amazon EC2 is a security mechanism used to
 authenticate and secure
            access to your EC2 instances. It consists of a public key and a private
 key,
            which are generated as a pair.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<CreateKeyPairResponse> future =
 ec2Actions.createKeyPairAsync(keyName, fileName);
            CreateKeyPairResponse response = future.join();
            logger.info("Key Pair successfully created. Key Fingerprint: " +
 response.keyFingerprint());

        } catch (RuntimeException rt) {
```

```
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                if (ec2Ex.getMessage().contains("already exists")) {
                    // Key pair already exists.
                    logger.info("The key pair '" + keyName + "' already exists.
Moving on...");
                } else {
                    logger.info("EC2 error occurred: Error message: {}, Error code
{}", ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                    return;
                }
            } else {
                logger.info("An unexpected error occurred: " + (rt.getMessage()));
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. List key pairs.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<DescribeKeyPairsResponse> future =
ec2Actions.describeKeysAsync();
            DescribeKeyPairsResponse keyPairsResponse = future.join();
            keyPairsResponse.keyPairs().forEach(keyPair -> logger.info(
                "Found key pair with name {} and fingerprint {}",
                keyPair.keyName(),
                keyPair.keyFingerprint()));

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Error message: {}, Error code {}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", (cause != null ?
cause.getMessage() : rt.getMessage()));
                return;
            }
        }
        waitForInputToContinue(scanner);
```

```
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Create a security group.");
        logger.info("""
            An AWS EC2 Security Group is a virtual firewall that controls the
            inbound and outbound traffic to an EC2 instance. It acts as a first
line
            of defense for your EC2 instances, allowing you to specify the rules
that
            govern the network traffic entering and leaving your instances.
           """);
        waitForInputToContinue(scanner);
        String groupId = "";
        try {
            CompletableFuture<String> future =
ec2Actions.createSecurityGroupAsync(groupName, groupDesc, vpcId, myIpAddress);
            future.join();
            logger.info("Created security group") ;

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                if (ec2Ex.awsErrorDetails().errorMessage().contains("already
exists")) {
                    logger.info("The Security Group already exists. Moving on...");
                } else {
                    logger.error("An unexpected error occurred: {}",
ec2Ex.awsErrorDetails().errorMessage());
                    return;
                }
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Display security group information for the new security
group.");
        waitForInputToContinue(scanner);
```

```
        try {
            CompletableFuture<String> future =
ec2Actions.describeSecurityGroupArnByNameAsync(groupName);
            groupId = future.join();
            logger.info("The security group Id is "+groupId);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                String errorCode = ec2Ex.awsErrorDetails().errorCode();
                if ("InvalidGroup.NotFound".equals(errorCode)) {
                    logger.info("Security group '{}' does not exist. Error Code:
{}", groupName, errorCode);
                } else {
                    logger.info("EC2 error occurred: Message {}, Error Code: {}",
ec2Ex.getMessage(), errorCode);
                }
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Get a list of Amazon Linux 2 AMIs and select one with amzn2
in the name.");
        logger.info("""
            An Amazon EC2 AMI (Amazon Machine Image) is a pre-configured virtual
machine image that
            serves as a template for launching EC2 instances. It contains all the
necessary software and
            configurations required to run an application or operating system on an
EC2 instance.
            """);
        waitForInputToContinue(scanner);
        String instanceAMI="";
        try {
            CompletableFuture<GetParametersByPathResponse> future =
ec2Actions.getParaValuesAsync();
            GetParametersByPathResponse pathResponse = future.join();
            List<Parameter> parameterList = pathResponse.parameters();
            for (Parameter para : parameterList) {
                if (filterName(para.name())) {
```

```
                    instanceAMI = para.value();
                    break;
                }
            }
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        logger.info("The AMI value with amzn2 is: {}", instanceAMI);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. Get the (Amazon Machine Image) AMI value from the amzn2
image.");
        logger.info("""
           An AMI value represents a specific version of a virtual machine (VM) or
server image.
           It uniquely identifies a particular version of an EC2 instance, including
its operating system,
           pre-installed software, and any custom configurations. This allows you to
consistently deploy the same
           VM image across your infrastructure.

            """);
        waitForInputToContinue(scanner);
        String amiValue;
        try {
            CompletableFuture<String> future =
ec2Actions.describeImageAsync(instanceAMI);
            amiValue = future.join();

        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof Ec2Exception) {
                Ec2Exception ec2Ex = (Ec2Exception) cause;
```

```
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Retrieves an instance type available in the current AWS
region.");
        waitForInputToContinue(scanner);
        String instanceType;
        try {
            CompletableFuture<String> future = ec2Actions.getInstanceTypesAsync();
            instanceType = future.join();
            if (!instanceType.isEmpty()) {
                logger.info("Found instance type: " + instanceType);
            } else {
                logger.info("Desired instance type not found.");
            }
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Create an Amazon EC2 instance using the key pair, the
instance type, the security group, and the EC2 AMI value.");
        logger.info("Once the EC2 instance is created, it is placed into a running
state.");
        waitForInputToContinue(scanner);
```

```
        String newInstanceId;
        try {
            CompletableFuture<String> future =
ec2Actions.runInstanceAsync(instanceType, keyName, groupName, amiValue);
            newInstanceId = future.join();
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception) {
                Ec2Exception ec2Ex = (Ec2Exception) cause;
                switch (ec2Ex.awsErrorDetails().errorCode()) {
                    case "InvalidParameterValue":
                        logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                        break;
                    case "InsufficientInstanceCapacity":
                        // Handle insufficient instance capacity.
                        logger.info("Insufficient instance capacity: {}, {}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                        break;
                    case "InvalidGroup.NotFound":
                        // Handle security group not found.
                        logger.info("Security group not found: {},{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                        break;
                    default:
                        logger.info("EC2 error occurred: {} (Code: {}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                        break;
                }
                return;
            } else {
                logger.info("An unexpected error occurred: {}", (cause != null ?
cause.getMessage() : rt.getMessage()));
                return;
            }
        }
        logger.info("The instance Id is " + newInstanceId);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("9. Display information about the running instance. ");

        waitForInputToContinue(scanner);
```

```
        String publicIp;
        try {
            CompletableFuture<String> future =
ec2Actions.describeEC2InstancesAsync(newInstanceId);
            publicIp = future.join();
            logger.info("EC2 instance public IP {}", publicIp);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }

        }
        logger.info("You can SSH to the instance using this command:");
        logger.info("ssh -i " + fileName + " ec2-user@" + publicIp);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("10. Stop the instance using a waiter (this may take a few
mins).");
        // Remove the 2nd one
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
ec2Actions.stopInstanceAsync(newInstanceId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
```

```
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("11. Start the instance using a waiter (this may take a few
mins).");
        try {
            CompletableFuture<Void> future =
ec2Actions.startInstanceAsync(newInstanceId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                // Handle EC2 exceptions.
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("12. Allocate an Elastic IP address and associate it with the
instance.");
        logger.info("""
            An Elastic IP address is a static public IP address that you can
associate with your EC2 instance.
            This allows you to have a fixed, predictable IP address that remains the
same even if your instance
            is stopped, terminated, or replaced.
            This is particularly useful for applications or services that need to be
accessed consistently from a
            known IP address.

            An EC2 Allocation ID (also known as a Reserved Instance Allocation
ID) is a unique identifier associated with a Reserved Instance (RI) that you have
purchased in AWS.
```

```
            When you purchase a Reserved Instance, AWS assigns a unique Allocation
ID to it.
            This Allocation ID is used to track and identify the specific RI you
have purchased,
            and it is important for managing and monitoring your Reserved Instances.

            """);

        waitForInputToContinue(scanner);
        String allocationId;
        try {
            CompletableFuture<String> future = ec2Actions.allocateAddressAsync();
            allocationId = future.join();
            logger.info("Successfully allocated address with ID: " +allocationId);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        logger.info("The allocation Id value is " + allocationId);
        waitForInputToContinue(scanner);
        String associationId;
        try {
            CompletableFuture<String> future =
ec2Actions.associateAddressAsync(newInstanceId, allocationId);
            associationId = future.join();
            logger.info("Successfully associated address with ID: " +associationId);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
```

```
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("13. Describe the instance again. Note that the public IP
address has changed");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
ec2Actions.describeEC2InstancesAsync(newInstanceId);
            publicIp = future.join();
            logger.info("EC2 instance public IP: " + publicIp);
            logger.info("You can SSH to the instance using this command:");
            logger.info("ssh -i " + fileName + " ec2-user@" + publicIp);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("14. Disassociate and release the Elastic IP address.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<DisassociateAddressResponse> future =
ec2Actions.disassociateAddressAsync(associationId);
            future.join();
            logger.info("Address successfully disassociated.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                // Handle EC2 exceptions.
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
```

```
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<ReleaseAddressResponse> future =
ec2Actions.releaseEC2AddressAsync(allocationId);
            future.join(); // Wait for the operation to complete
            logger.info("Elastic IP address successfully released.");
        } catch (RuntimeException rte) {
            logger.info("An unexpected error occurred: {}", rte.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("15. Terminate the instance and use a waiter (this may take a
few mins).");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Object> future =
ec2Actions.terminateEC2Async(newInstanceId);
            future.join();
            logger.info("EC2 instance successfully terminated.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                // Handle EC2 exceptions.
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("16. Delete the security group.");
        waitForInputToContinue(scanner);
        try {
```

```
            CompletableFuture<Void> future =
ec2Actions.deleteEC2SecGroupAsync(groupId);
            future.join();
            logger.info("Security group successfully deleted.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("17. Delete the key.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<DeleteKeyPairResponse> future =
ec2Actions.deleteKeysAsync(keyName);
            future.join();
            logger.info("Successfully deleted key pair named " + keyName);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage());
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("You successfully completed the Amazon EC2 scenario.");
        logger.info(DASHES);
    }
```

```
    public static boolean filterName(String name) {
        String[] parts = name.split("/");
        String myValue = parts[4];
        return myValue.contains("amzn2");
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

定义一个封装 EC2 动作的类。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.AllocateAddressRequest;
import software.amazon.awssdk.services.ec2.model.AllocateAddressResponse;
import software.amazon.awssdk.services.ec2.model.AssociateAddressRequest;
import software.amazon.awssdk.services.ec2.model.AssociateAddressResponse;
import
 software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressRequest;
import software.amazon.awssdk.services.ec2.model.CreateKeyPairRequest;
import software.amazon.awssdk.services.ec2.model.CreateKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupRequest;
```

```
import software.amazon.awssdk.services.ec2.model.DeleteKeyPairRequest;
import software.amazon.awssdk.services.ec2.model.DeleteKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.DeleteSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.DeleteSecurityGroupResponse;
import software.amazon.awssdk.services.ec2.model.DescribeImagesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstanceTypesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstanceTypesResponse;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeKeyPairsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest;
import software.amazon.awssdk.services.ec2.model.DisassociateAddressRequest;
import software.amazon.awssdk.services.ec2.model.DisassociateAddressResponse;
import software.amazon.awssdk.services.ec2.model.DomainType;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.Filter;
import software.amazon.awssdk.services.ec2.model.InstanceTypeInfo;
import software.amazon.awssdk.services.ec2.model.IpPermission;
import software.amazon.awssdk.services.ec2.model.IpRange;
import software.amazon.awssdk.services.ec2.model.ReleaseAddressRequest;
import software.amazon.awssdk.services.ec2.model.ReleaseAddressResponse;
import software.amazon.awssdk.services.ec2.model.RunInstancesRequest;
import software.amazon.awssdk.services.ec2.model.RunInstancesResponse;
import software.amazon.awssdk.services.ec2.model.StopInstancesRequest;
import software.amazon.awssdk.services.ec2.model.StartInstancesRequest;
import software.amazon.awssdk.services.ec2.model.TerminateInstancesRequest;
import software.amazon.awssdk.services.ec2.model.Vpc;
import software.amazon.awssdk.services.ec2.paginators.DescribeImagesPublisher;
import software.amazon.awssdk.services.ec2.paginators.DescribeInstancesPublisher;
import
 software.amazon.awssdk.services.ec2.paginators.DescribeSecurityGroupsPublisher;
import software.amazon.awssdk.services.ec2.paginators.DescribeVpcsPublisher;
import software.amazon.awssdk.services.ec2.waiters.Ec2AsyncWaiter;
import software.amazon.awssdk.services.ssm.SsmAsyncClient;
import software.amazon.awssdk.services.ssm.model.GetParametersByPathRequest;
import software.amazon.awssdk.services.ssm.model.GetParametersByPathResponse;
import software.amazon.awssdk.services.ec2.model.TerminateInstancesResponse;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
```

```
import java.util.concurrent.CompletionException;
import java.util.concurrent.atomic.AtomicReference;

public class EC2Actions {
    private static final Logger logger = LoggerFactory.getLogger(EC2Actions.class);
    private static Ec2AsyncClient ec2AsyncClient;

    /**
     * Retrieves an asynchronous Amazon Elastic Container Registry (ECR) client.
     *
     * @return the configured ECR asynchronous client.
     */
    private static Ec2AsyncClient getAsyncClient() {
        if (ec2AsyncClient == null) {
            /*
            The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
 version 2,
            and it is designed to provide a high-performance, asynchronous HTTP
 client for interacting with AWS services.
             It uses the Netty framework to handle the underlying network
 communication and the Java NIO API to
             provide a non-blocking, event-driven approach to HTTP requests and
 responses.
             */
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(50)  // Adjust as needed.
                .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
 timeout.
                .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
                .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
               .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
 timeout.
                .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
 individual call attempt timeout.
                .build();

            ec2AsyncClient = Ec2AsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
```

```
                .build();
        }
        return ec2AsyncClient;
    }

    /**
     * Deletes a key pair asynchronously.
     *
     * @param keyPair the name of the key pair to delete
     * @return a {@link CompletableFuture} that represents the result of the
asynchronous operation.
     *         The {@link CompletableFuture} will complete with a {@link
DeleteKeyPairResponse} object
     *         that provides the result of the key pair deletion operation.
     */
    public CompletableFuture<DeleteKeyPairResponse> deleteKeysAsync(String keyPair)
{
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
            .keyName(keyPair)
            .build();

        // Initiate the asynchronous request to delete the key pair.
        CompletableFuture<DeleteKeyPairResponse> response =
getAsyncClient().deleteKeyPair(request);
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete key pair: " + keyPair,
ex);
            } else if (resp == null) {
                throw new RuntimeException("No response received for deleting key
pair: " + keyPair);
            }
        });
    }

    /**
     * Deletes an EC2 security group asynchronously.
     *
     * @param groupId the ID of the security group to delete
     * @return a CompletableFuture that completes when the security group is deleted
     */
    public CompletableFuture<Void> deleteEC2SecGroupAsync(String groupId) {
        DeleteSecurityGroupRequest request = DeleteSecurityGroupRequest.builder()
            .groupId(groupId)
```

```
            .build();

        CompletableFuture<DeleteSecurityGroupResponse> response =
getAsyncClient().deleteSecurityGroup(request);
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete security group with Id
" + groupId, ex);
            } else if (resp == null) {
                throw new RuntimeException("No response received for deleting
security group with Id " + groupId);
            }
        }).thenApply(resp -> null);
    }

    /**
     * Terminates an EC2 instance asynchronously and waits for it to reach the
terminated state.
     *
     * @param instanceId the ID of the EC2 instance to terminate
     * @return a {@link CompletableFuture} that completes when the instance has been
terminated
     * @throws RuntimeException if there is no response from the AWS SDK or if there
is a failure during the termination process
     */
    public CompletableFuture<Object> terminateEC2Async(String instanceId) {
        TerminateInstancesRequest terminateRequest =
TerminateInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        CompletableFuture<TerminateInstancesResponse> responseFuture =
getAsyncClient().terminateInstances(terminateRequest);
        return responseFuture.thenCompose(terminateResponse -> {
            if (terminateResponse == null) {
                throw new RuntimeException("No response received for terminating
instance " + instanceId);
            }
            System.out.println("Going to terminate an EC2 instance and use a waiter
to wait for it to be in terminated state");
            return getAsyncClient().waiter()
                .waitUntilInstanceTerminated(r -> r.instanceIds(instanceId))
                .thenApply(waiterResponse -> null);
        }).exceptionally(throwable -> {
```

```
            // Handle any exceptions that occurred during the async call
            throw new RuntimeException("Failed to terminate EC2 instance: " +
throwable.getMessage(), throwable);
        });
    }

    /**
     * Releases an Elastic IP address asynchronously.
     *
     * @param allocId the allocation ID of the Elastic IP address to be released
     * @return a {@link CompletableFuture} representing the asynchronous operation
of releasing the Elastic IP address
     */
    public CompletableFuture<ReleaseAddressResponse> releaseEC2AddressAsync(String
allocId) {
        ReleaseAddressRequest request = ReleaseAddressRequest.builder()
            .allocationId(allocId)
            .build();

        CompletableFuture<ReleaseAddressResponse> response =
getAsyncClient().releaseAddress(request);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to release Elastic IP address",
ex);
            }
        });

        return response;
    }

    /**
     * Disassociates an Elastic IP address from an instance asynchronously.
     *
     * @param associationId The ID of the association you want to disassociate.
     * @return a {@link CompletableFuture} representing the asynchronous operation
of disassociating the address. The
     *         {@link CompletableFuture} will complete with a {@link
DisassociateAddressResponse} when the operation is
     *         finished.
     * @throws RuntimeException if the disassociation of the address fails.
     */
    public CompletableFuture<DisassociateAddressResponse>
disassociateAddressAsync(String associationId) {
```

```
        Ec2AsyncClient ec2 = getAsyncClient();
        DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
            .associationId(associationId)
            .build();

        // Disassociate the address asynchronously.
        CompletableFuture<DisassociateAddressResponse> response =
ec2.disassociateAddress(addressRequest);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
               throw new RuntimeException("Failed to disassociate address", ex);
            }
        });

        return response;
    }

    /**
     * Associates an Elastic IP address with an EC2 instance asynchronously.
     *
     * @param instanceId    the ID of the EC2 instance to associate the Elastic IP
address with
     * @param allocationId  the allocation ID of the Elastic IP address to associate
     * @return a {@link CompletableFuture} that completes with the association ID
when the operation is successful,
     *         or throws a {@link RuntimeException} if the operation fails
     */
    public CompletableFuture<String> associateAddressAsync(String instanceId, String
allocationId) {
        AssociateAddressRequest associateRequest = AssociateAddressRequest.builder()
            .instanceId(instanceId)
            .allocationId(allocationId)
            .build();

        CompletableFuture<AssociateAddressResponse> responseFuture =
getAsyncClient().associateAddress(associateRequest);
        return responseFuture.thenApply(response -> {
            if (response.associationId() != null) {
                return response.associationId();
            } else {
                throw new RuntimeException("Association ID is null after associating
address.");
            }
```

```
        }).whenComplete((result, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to associate address", ex);
            }
        });
    }

    /**
     * Allocates an Elastic IP address asynchronously in the VPC domain.
     *
     * @return a {@link CompletableFuture} containing the allocation ID of the
allocated Elastic IP address
     */
    public CompletableFuture<String> allocateAddressAsync() {
        AllocateAddressRequest allocateRequest = AllocateAddressRequest.builder()
            .domain(DomainType.VPC)
            .build();

        CompletableFuture<AllocateAddressResponse> responseFuture =
getAsyncClient().allocateAddress(allocateRequest);
        return
responseFuture.thenApply(AllocateAddressResponse::allocationId).whenComplete((result,
ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to allocate address", ex);
            }
        });
    }

    /**
     * Asynchronously describes the state of an EC2 instance.
     * The paginator helps you iterate over multiple pages of results.
     *
     * @param newInstanceId the ID of the EC2 instance to describe
     * @return a {@link CompletableFuture} that, when completed, contains a string
describing the state of the EC2 instance
     */
    public CompletableFuture<String> describeEC2InstancesAsync(String newInstanceId)
{
        DescribeInstancesRequest request = DescribeInstancesRequest.builder()
            .instanceIds(newInstanceId)
            .build();
```

```
        DescribeInstancesPublisher paginator =
getAsyncClient().describeInstancesPaginator(request);
        AtomicReference<String> publicIpAddressRef = new AtomicReference<>();
        return paginator.subscribe(response -> {
            response.reservations().stream()
                .flatMap(reservation -> reservation.instances().stream())
                .filter(instance -> instance.instanceId().equals(newInstanceId))
                .findFirst()
                .ifPresent(instance ->
publicIpAddressRef.set(instance.publicIpAddress()));
        }).thenApply(v -> {
            String publicIpAddress = publicIpAddressRef.get();
            if (publicIpAddress == null) {
                throw new RuntimeException("Instance with ID " + newInstanceId + "
not found.");
            }
            return publicIpAddress;
        }).exceptionally(ex -> {
            logger.info("Failed to describe instances: " + ex.getMessage());
            throw new RuntimeException("Failed to describe instances", ex);
        });
    }

    /**
     * Runs an EC2 instance asynchronously.
     *
     * @param instanceType The instance type to use for the EC2 instance.
     * @param keyName The name of the key pair to associate with the EC2 instance.
     * @param groupName The name of the security group to associate with the EC2
instance.
     * @param amiId The ID of the Amazon Machine Image (AMI) to use for the EC2
instance.
     * @return A {@link CompletableFuture} that completes with the ID of the started
EC2 instance.
     * @throws RuntimeException If there is an error running the EC2 instance.
     */
    public CompletableFuture<String> runInstanceAsync(String instanceType, String
keyName, String groupName, String amiId) {
        RunInstancesRequest runRequest = RunInstancesRequest.builder()
            .instanceType(instanceType)
            .keyName(keyName)
            .securityGroups(groupName)
            .maxCount(1)
            .minCount(1)
```

```
            .imageId(amiId)
            .build();

        CompletableFuture<RunInstancesResponse> responseFuture =
getAsyncClient().runInstances(runRequest);
        return responseFuture.thenCompose(response -> {
            String instanceIdVal = response.instances().get(0).instanceId();
            System.out.println("Going to start an EC2 instance and use a waiter to
wait for it to be in running state");
            return getAsyncClient().waiter()
                .waitUntilInstanceExists(r -> r.instanceIds(instanceIdVal))
                .thenCompose(waitResponse -> getAsyncClient().waiter()
                    .waitUntilInstanceRunning(r -> r.instanceIds(instanceIdVal))
                    .thenApply(runningResponse -> instanceIdVal));
        }).exceptionally(throwable -> {
            // Handle any exceptions that occurred during the async call
            throw new RuntimeException("Failed to run EC2 instance: " +
throwable.getMessage(), throwable);
        });
    }

    /**
     * Asynchronously retrieves the instance types available in the current AWS
region.
     * <p>
     * This method uses the AWS SDK's asynchronous API to fetch the available
instance types
     * and then processes the response. It logs the memory information, network
information,
     * and instance type for each instance type returned. Additionally, it returns a
     * {@link CompletableFuture} that resolves to the instance type string for the
"t2.2xlarge"
     * instance type, if it is found in the response. If the "t2.2xlarge" instance
type is not
     * found, an empty string is returned.
     * </p>
     *
     * @return a {@link CompletableFuture} that resolves to the instance type string
for the
     * "t2.2xlarge" instance type, or an empty string if the instance type is not
found
     */
    public CompletableFuture<String> getInstanceTypesAsync() {
```

```
        DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
            .maxResults(10)
            .build();

        CompletableFuture<DescribeInstanceTypesResponse> response =
getAsyncClient().describeInstanceTypes(typesRequest);
        response.whenComplete((resp, ex) -> {
            if (resp != null) {
                List<InstanceTypeInfo> instanceTypes = resp.instanceTypes();
                for (InstanceTypeInfo type : instanceTypes) {
                    logger.info("The memory information of this type is " +
type.memoryInfo().sizeInMiB());
                    logger.info("Network information is " +
type.networkInfo().toString());
                    logger.info("Instance type is " +
type.instanceType().toString());
                }
            } else {
                throw (RuntimeException) ex;
            }
        });

        return response.thenApply(resp -> {
            for (InstanceTypeInfo type : resp.instanceTypes()) {
                String instanceType = type.instanceType().toString();
                if (instanceType.equals("t2.2xlarge")) {
                    return instanceType;
                }
            }
            return "";
        });
    }

    /**
     * Asynchronously describes an AWS EC2 image with the specified image ID.
     *
     * @param imageId the ID of the image to be described
     * @return a {@link CompletableFuture} that, when completed, contains the ID of
the described image
     * @throws RuntimeException if no images are found with the provided image ID,
or if an error occurs during the AWS API call
     */
    public CompletableFuture<String> describeImageAsync(String imageId) {
```

```
        DescribeImagesRequest imagesRequest = DescribeImagesRequest.builder()
            .imageIds(imageId)
            .build();

        AtomicReference<String> imageIdRef = new AtomicReference<>();
        DescribeImagesPublisher paginator =
getAsyncClient().describeImagesPaginator(imagesRequest);
        return paginator.subscribe(response -> {
            response.images().stream()
                .filter(image -> image.imageId().equals(imageId))
                .findFirst()
                .ifPresent(image -> {
                    logger.info("The description of the image is " +
image.description());
                    logger.info("The name of the image is " + image.name());
                    imageIdRef.set(image.imageId());
                });
        }).thenApply(v -> {
            String id = imageIdRef.get();
            if (id == null) {
                throw new RuntimeException("No images found with the provided image
ID.");
            }
            return id;
        }).exceptionally(ex -> {
            logger.info("Failed to describe image: " + ex.getMessage());
            throw new RuntimeException("Failed to describe image", ex);
        });
    }

    /**
     * Retrieves the parameter values asynchronously using the AWS Systems Manager
(SSM) API.
     *
     * @return a {@link CompletableFuture} that holds the response from the SSM API
call to get parameters by path
     */
    public CompletableFuture<GetParametersByPathResponse> getParaValuesAsync() {
        SsmAsyncClient ssmClient = SsmAsyncClient.builder()
            .region(Region.US_EAST_1)
            .build();

        GetParametersByPathRequest parameterRequest =
GetParametersByPathRequest.builder()
```

```
            .path("/aws/service/ami-amazon-linux-latest")
            .build();

        // Create a CompletableFuture to hold the final result.
        CompletableFuture<GetParametersByPathResponse> responseFuture = new
CompletableFuture<>();
        ssmClient.getParametersByPath(parameterRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    responseFuture.completeExceptionally(new
RuntimeException("Failed to get parameters by path", exception));
                } else {
                    responseFuture.complete(response);
                }
            });

        return responseFuture;
    }


    /**
     * Asynchronously describes the security groups for the specified group ID.
     *
     * @param groupName the name of the security group to describe
     * @return a {@link CompletableFuture} that represents the asynchronous
operation
     *         of describing the security groups. The future will complete with a
     *         {@link DescribeSecurityGroupsResponse} object that contains the
     *         security group information.
     */
    public CompletableFuture<String> describeSecurityGroupArnByNameAsync(String
groupName) {
        DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
            .groupNames(groupName)
            .build();

        DescribeSecurityGroupsPublisher paginator =
getAsyncClient().describeSecurityGroupsPaginator(request);
        AtomicReference<String> groupIdRef = new AtomicReference<>();
        return paginator.subscribe(response -> {
            response.securityGroups().stream()
                .filter(securityGroup ->
securityGroup.groupName().equals(groupName))
```

```
                .findFirst()
                .ifPresent(securityGroup ->
groupIdRef.set(securityGroup.groupId()));
        }).thenApply(v -> {
            String groupId = groupIdRef.get();
            if (groupId == null) {
                throw new RuntimeException("No security group found with the name: "
+ groupName);
            }
            return groupId;
        }).exceptionally(ex -> {
            logger.info("Failed to describe security group: " + ex.getMessage());
            throw new RuntimeException("Failed to describe security group", ex);
        });
    }

    /**
     * Creates a new security group asynchronously with the specified group name,
description, and VPC ID. It also
     * authorizes inbound traffic on ports 80 and 22 from the specified IP address.
     *
     * @param groupName    the name of the security group to create
     * @param groupDesc    the description of the security group
     * @param vpcId        the ID of the VPC in which to create the security group
     * @param myIpAddress  the IP address from which to allow inbound traffic (e.g.,
"192.168.1.1/0" to allow traffic from
     *                     any IP address in the 192.168.1.0/24 subnet)
     * @return a CompletableFuture that, when completed, returns the ID of the
created security group
     * @throws RuntimeException if there was a failure creating the security group
or authorizing the inbound traffic
     */
    public CompletableFuture<String> createSecurityGroupAsync(String groupName,
String groupDesc, String vpcId, String myIpAddress) {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

        return getAsyncClient().createSecurityGroup(createRequest)
            .thenCompose(createResponse -> {
                String groupId = createResponse.groupId();
```

```
                IpRange ipRange = IpRange.builder()
                    .cidrIp(myIpAddress + "/32")
                    .build();

                IpPermission ipPerm = IpPermission.builder()
                    .ipProtocol("tcp")
                    .toPort(80)
                    .fromPort(80)
                    .ipRanges(ipRange)
                    .build();

                IpPermission ipPerm2 = IpPermission.builder()
                    .ipProtocol("tcp")
                    .toPort(22)
                    .fromPort(22)
                    .ipRanges(ipRange)
                    .build();

                AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                    .groupName(groupName)
                    .ipPermissions(ipPerm, ipPerm2)
                    .build();

                return getAsyncClient().authorizeSecurityGroupIngress(authRequest)
                    .thenApply(authResponse -> groupId);
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    if (exception instanceof CompletionException &&
exception.getCause() instanceof Ec2Exception) {
                        throw (Ec2Exception) exception.getCause();
                    } else {
                        throw new RuntimeException("Failed to create security group:
" + exception.getMessage(), exception);
                    }
                }
            });
    }

    /**
     * Asynchronously describes the key pairs associated with the current AWS
account.
     *
```

```
    * @return a {@link CompletableFuture} containing the {@link
DescribeKeyPairsResponse} object, which provides
    * information about the key pairs.
    */
   public CompletableFuture<DescribeKeyPairsResponse> describeKeysAsync() {
       CompletableFuture<DescribeKeyPairsResponse> responseFuture =
getAsyncClient().describeKeyPairs();
       responseFuture.whenComplete((response, exception) -> {
           if (exception != null) {
             throw new RuntimeException("Failed to describe key pairs: " +
exception.getMessage(), exception);
           }
       });

       return responseFuture;
   }

   /**
    * Creates a new key pair asynchronously.
    *
    * @param keyName the name of the key pair to create
    * @param fileName the name of the file to write the key material to
    * @return a {@link CompletableFuture} that represents the asynchronous
operation
    *         of creating the key pair and writing the key material to a file
    */
   public CompletableFuture<CreateKeyPairResponse> createKeyPairAsync(String
keyName, String fileName) {
       CreateKeyPairRequest request = CreateKeyPairRequest.builder()
           .keyName(keyName)
           .build();

       CompletableFuture<CreateKeyPairResponse> responseFuture =
getAsyncClient().createKeyPair(request);
       responseFuture.whenComplete((response, exception) -> {
           if (response != null) {
               try {
                   BufferedWriter writer = new BufferedWriter(new
FileWriter(fileName));
                   writer.write(response.keyMaterial());
                   writer.close();
               } catch (IOException e) {
                   throw new RuntimeException("Failed to write key material to
file: " + e.getMessage(), e);
```

```
                }
            } else {
                throw new RuntimeException("Failed to create key pair: " +
exception.getMessage(), exception);
            }
        });

        return responseFuture;
    }

    /**
     * Describes the first default VPC asynchronously and using a paginator.
     *
     * @return a {@link CompletableFuture} that, when completed, contains the first
default VPC found.\
     */
    public CompletableFuture<Vpc> describeFirstEC2VpcAsync() {
        Filter myFilter = Filter.builder()
            .name("is-default")
            .values("true")
            .build();

        DescribeVpcsRequest request = DescribeVpcsRequest.builder()
            .filters(myFilter)
            .build();

        DescribeVpcsPublisher paginator =
getAsyncClient().describeVpcsPaginator(request);
        AtomicReference<Vpc> vpcRef = new AtomicReference<>();
        return paginator.subscribe(response -> {
            response.vpcs().stream()
                .findFirst()
                .ifPresent(vpcRef::set);
        }).thenApply(v -> {
            Vpc vpc = vpcRef.get();
            if (vpc == null) {
                throw new RuntimeException("Default VPC not found");
            }
            return vpc;
        }).exceptionally(ex -> {
            logger.info("Failed to describe VPCs: " + ex.getMessage());
            throw new RuntimeException("Failed to describe VPCs", ex);
        });
    }
```

```
   /**
    * Stops the EC2 instance with the specified ID asynchronously and waits for the
instance to stop.
    *
    * @param instanceId the ID of the EC2 instance to stop
    * @return a {@link CompletableFuture} that completes when the instance has been
stopped, or exceptionally if an error occurs
    */
   public CompletableFuture<Void> stopInstanceAsync(String instanceId) {
       StopInstancesRequest stopRequest = StopInstancesRequest.builder()
           .instanceIds(instanceId)
           .build();

       DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
           .instanceIds(instanceId)
           .build();

       Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
           .client(getAsyncClient())
           .build();

       CompletableFuture<Void> resultFuture = new CompletableFuture<>();
       logger.info("Stopping instance " + instanceId + " and waiting for it to
stop.");
       getAsyncClient().stopInstances(stopRequest)
           .thenCompose(response -> {
               if (response.stoppingInstances().isEmpty()) {
                   return CompletableFuture.failedFuture(new RuntimeException("No
instances were stopped. Please check the instance ID: " + instanceId));
               }
               return ec2Waiter.waitUntilInstanceStopped(describeRequest);
           })
           .thenAccept(waiterResponse -> {
               logger.info("Successfully stopped instance " + instanceId);
               resultFuture.complete(null);
           })
           .exceptionally(throwable -> {
               logger.error("Failed to stop instance " + instanceId + ": " +
throwable.getMessage(), throwable);
               resultFuture.completeExceptionally(new RuntimeException("Failed to
stop instance: " + throwable.getMessage(), throwable));
               return null;
```

```
            });

        return resultFuture;
    }

    /**
     * Starts an Amazon EC2 instance asynchronously and waits until it is in the
"running" state.
     *
     * @param instanceId the ID of the instance to start
     * @return a {@link CompletableFuture} that completes when the instance has been
started and is in the "running" state, or exceptionally if an error occurs
     */
    public CompletableFuture<Void> startInstanceAsync(String instanceId) {
        StartInstancesRequest startRequest = StartInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
            .client(getAsyncClient())
            .build();

        DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        logger.info("Starting instance " + instanceId + " and waiting for it to
run.");
        CompletableFuture<Void> resultFuture = new CompletableFuture<>();
        return getAsyncClient().startInstances(startRequest)
            .thenCompose(response ->
                ec2Waiter.waitUntilInstanceRunning(describeRequest)
            )
            .thenAccept(waiterResponse -> {
                logger.info("Successfully started instance " + instanceId);
                resultFuture.complete(null);
            })
            .exceptionally(throwable -> {
                resultFuture.completeExceptionally(new RuntimeException("Failed to
start instance: " + throwable.getMessage(), throwable));
                return null;
            });
    }
```

```
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》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

操作

`AllocateAddress`

以下代码示例演示了如何使用 `AllocateAddress`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Allocates an Elastic IP address asynchronously in the VPC domain.
     *
     * @return a {@link CompletableFuture} containing the allocation ID of the
allocated Elastic IP address
     */
    public CompletableFuture<String> allocateAddressAsync() {
        AllocateAddressRequest allocateRequest = AllocateAddressRequest.builder()
            .domain(DomainType.VPC)
            .build();

        CompletableFuture<AllocateAddressResponse> responseFuture =
getAsyncClient().allocateAddress(allocateRequest);
        return
responseFuture.thenApply(AllocateAddressResponse::allocationId).whenComplete((result,
ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to allocate address", ex);
            }
        });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考AllocateAddress中的。

`AssociateAddress`

以下代码示例演示了如何使用 `AssociateAddress`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Associates an Elastic IP address with an EC2 instance asynchronously.
    *
    * @param instanceId    the ID of the EC2 instance to associate the Elastic IP
address with
    * @param allocationId  the allocation ID of the Elastic IP address to associate
    * @return a {@link CompletableFuture} that completes with the association ID
when the operation is successful,
    *         or throws a {@link RuntimeException} if the operation fails
    */
   public CompletableFuture<String> associateAddressAsync(String instanceId, String
allocationId) {
       AssociateAddressRequest associateRequest = AssociateAddressRequest.builder()
           .instanceId(instanceId)
           .allocationId(allocationId)
           .build();

       CompletableFuture<AssociateAddressResponse> responseFuture =
getAsyncClient().associateAddress(associateRequest);
       return responseFuture.thenApply(response -> {
           if (response.associationId() != null) {
               return response.associationId();
           } else {
               throw new RuntimeException("Association ID is null after associating
address.");
           }
       }).whenComplete((result, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to associate address", ex);
           }
       });
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考AssociateAddress中的。

`AuthorizeSecurityGroupIngress`

以下代码示例演示了如何使用 `AuthorizeSecurityGroupIngress`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Creates a new security group asynchronously with the specified group name,
description, and VPC ID. It also
    * authorizes inbound traffic on ports 80 and 22 from the specified IP address.
    *
    * @param groupName    the name of the security group to create
    * @param groupDesc    the description of the security group
    * @param vpcId        the ID of the VPC in which to create the security group
    * @param myIpAddress  the IP address from which to allow inbound traffic (e.g.,
"192.168.1.1/0" to allow traffic from
    *                     any IP address in the 192.168.1.0/24 subnet)
    * @return a CompletableFuture that, when completed, returns the ID of the
created security group
    * @throws RuntimeException if there was a failure creating the security group
or authorizing the inbound traffic
    */
   public CompletableFuture<String> createSecurityGroupAsync(String groupName,
String groupDesc, String vpcId, String myIpAddress) {
       CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
           .groupName(groupName)
           .description(groupDesc)
           .vpcId(vpcId)
           .build();

       return getAsyncClient().createSecurityGroup(createRequest)
           .thenCompose(createResponse -> {
               String groupId = createResponse.groupId();
```

```
                IpRange ipRange = IpRange.builder()
                    .cidrIp(myIpAddress + "/32")
                    .build();

                IpPermission ipPerm = IpPermission.builder()
                    .ipProtocol("tcp")
                    .toPort(80)
                    .fromPort(80)
                    .ipRanges(ipRange)
                    .build();

                IpPermission ipPerm2 = IpPermission.builder()
                    .ipProtocol("tcp")
                    .toPort(22)
                    .fromPort(22)
                    .ipRanges(ipRange)
                    .build();

                AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                    .groupName(groupName)
                    .ipPermissions(ipPerm, ipPerm2)
                    .build();

                return getAsyncClient().authorizeSecurityGroupIngress(authRequest)
                    .thenApply(authResponse -> groupId);
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    if (exception instanceof CompletionException &&
exception.getCause() instanceof Ec2Exception) {
                        throw (Ec2Exception) exception.getCause();
                    } else {
                        throw new RuntimeException("Failed to create security group:
" + exception.getMessage(), exception);
                    }
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[AuthorizeSecurityGroupIngress](#)中的。

CreateKeyPair

以下代码示例演示了如何使用 CreateKeyPair。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Creates a new key pair asynchronously.
    *
    * @param keyName the name of the key pair to create
    * @param fileName the name of the file to write the key material to
    * @return a {@link CompletableFuture} that represents the asynchronous
operation
    *         of creating the key pair and writing the key material to a file
    */
   public CompletableFuture<CreateKeyPairResponse> createKeyPairAsync(String
keyName, String fileName) {
       CreateKeyPairRequest request = CreateKeyPairRequest.builder()
           .keyName(keyName)
           .build();

       CompletableFuture<CreateKeyPairResponse> responseFuture =
getAsyncClient().createKeyPair(request);
       responseFuture.whenComplete((response, exception) -> {
           if (response != null) {
               try {
                   BufferedWriter writer = new BufferedWriter(new
FileWriter(fileName));
                   writer.write(response.keyMaterial());
                   writer.close();
               } catch (IOException e) {
                   throw new RuntimeException("Failed to write key material to
file: " + e.getMessage(), e);
               }
           } else {
               throw new RuntimeException("Failed to create key pair: " +
exception.getMessage(), exception);
```

```
            }
        });

        return responseFuture;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateKeyPair中的。

`CreateSecurityGroup`

以下代码示例演示了如何使用 `CreateSecurityGroup`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Creates a new security group asynchronously with the specified group name,
description, and VPC ID. It also
     * authorizes inbound traffic on ports 80 and 22 from the specified IP address.
     *
     * @param groupName    the name of the security group to create
     * @param groupDesc    the description of the security group
     * @param vpcId        the ID of the VPC in which to create the security group
     * @param myIpAddress  the IP address from which to allow inbound traffic (e.g.,
"192.168.1.1/0" to allow traffic from
     *                     any IP address in the 192.168.1.0/24 subnet)
     * @return a CompletableFuture that, when completed, returns the ID of the
created security group
     * @throws RuntimeException if there was a failure creating the security group
or authorizing the inbound traffic
     */
    public CompletableFuture<String> createSecurityGroupAsync(String groupName,
String groupDesc, String vpcId, String myIpAddress) {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
```

```
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

        return getAsyncClient().createSecurityGroup(createRequest)
            .thenCompose(createResponse -> {
                String groupId = createResponse.groupId();
                IpRange ipRange = IpRange.builder()
                    .cidrIp(myIpAddress + "/32")
                    .build();

                IpPermission ipPerm = IpPermission.builder()
                    .ipProtocol("tcp")
                    .toPort(80)
                    .fromPort(80)
                    .ipRanges(ipRange)
                    .build();

                IpPermission ipPerm2 = IpPermission.builder()
                    .ipProtocol("tcp")
                    .toPort(22)
                    .fromPort(22)
                    .ipRanges(ipRange)
                    .build();

                AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                    .groupName(groupName)
                    .ipPermissions(ipPerm, ipPerm2)
                    .build();

                return getAsyncClient().authorizeSecurityGroupIngress(authRequest)
                    .thenApply(authResponse -> groupId);
            })
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    if (exception instanceof CompletionException &&
exception.getCause() instanceof Ec2Exception) {
                        throw (Ec2Exception) exception.getCause();
                    } else {
                        throw new RuntimeException("Failed to create security group:
" + exception.getMessage(), exception);
                    }
                }
```

```
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateSecurityGroup中的。

`DeleteKeyPair`

以下代码示例演示了如何使用 `DeleteKeyPair`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Deletes a key pair asynchronously.
     *
     * @param keyPair the name of the key pair to delete
     * @return a {@link CompletableFuture} that represents the result of the
asynchronous operation.
     *         The {@link CompletableFuture} will complete with a {@link
DeleteKeyPairResponse} object
     *         that provides the result of the key pair deletion operation.
     */
    public CompletableFuture<DeleteKeyPairResponse> deleteKeysAsync(String keyPair)
{
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
            .keyName(keyPair)
            .build();

        // Initiate the asynchronous request to delete the key pair.
        CompletableFuture<DeleteKeyPairResponse> response =
getAsyncClient().deleteKeyPair(request);
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete key pair: " + keyPair,
ex);
```

```
            } else if (resp == null) {
                throw new RuntimeException("No response received for deleting key
pair: " + keyPair);
            }
        });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteKeyPair中的。

`DeleteSecurityGroup`

以下代码示例演示了如何使用 `DeleteSecurityGroup`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Deletes an EC2 security group asynchronously.
     *
     * @param groupId the ID of the security group to delete
     * @return a CompletableFuture that completes when the security group is deleted
     */
    public CompletableFuture<Void> deleteEC2SecGroupAsync(String groupId) {
        DeleteSecurityGroupRequest request = DeleteSecurityGroupRequest.builder()
            .groupId(groupId)
            .build();

        CompletableFuture<DeleteSecurityGroupResponse> response =
getAsyncClient().deleteSecurityGroup(request);
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete security group with Id
" + groupId, ex);
            } else if (resp == null) {
                throw new RuntimeException("No response received for deleting
security group with Id " + groupId);
```

```
            }
        }).thenApply(resp -> null);
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteSecurityGroup中的。

`DescribeInstanceTypes`

以下代码示例演示了如何使用 `DescribeInstanceTypes`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Asynchronously retrieves the instance types available in the current AWS
region.
     * <p>
     * This method uses the AWS SDK's asynchronous API to fetch the available
instance types
     * and then processes the response. It logs the memory information, network
information,
     * and instance type for each instance type returned. Additionally, it returns a
     * {@link CompletableFuture} that resolves to the instance type string for the
"t2.2xlarge"
     * instance type, if it is found in the response. If the "t2.2xlarge" instance
type is not
     * found, an empty string is returned.
     * </p>
     *
     * @return a {@link CompletableFuture} that resolves to the instance type string
for the
     * "t2.2xlarge" instance type, or an empty string if the instance type is not
found
     */
    public CompletableFuture<String> getInstanceTypesAsync() {
```

```
        DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
            .maxResults(10)
            .build();

        CompletableFuture<DescribeInstanceTypesResponse> response =
getAsyncClient().describeInstanceTypes(typesRequest);
        response.whenComplete((resp, ex) -> {
            if (resp != null) {
                List<InstanceTypeInfo> instanceTypes = resp.instanceTypes();
                for (InstanceTypeInfo type : instanceTypes) {
                    logger.info("The memory information of this type is " +
type.memoryInfo().sizeInMiB());
                    logger.info("Network information is " +
type.networkInfo().toString());
                    logger.info("Instance type is " +
type.instanceType().toString());
                }
            } else {
                throw (RuntimeException) ex;
            }
        });

        return response.thenApply(resp -> {
            for (InstanceTypeInfo type : resp.instanceTypes()) {
                String instanceType = type.instanceType().toString();
                if (instanceType.equals("t2.2xlarge")) {
                    return instanceType;
                }
            }
            return "";
        });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeInstanceTypes中的。

`DescribeInstances`

以下代码示例演示了如何使用 `DescribeInstances`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Asynchronously describes the state of an EC2 instance.
     * The paginator helps you iterate over multiple pages of results.
     *
     * @param newInstanceId the ID of the EC2 instance to describe
     * @return a {@link CompletableFuture} that, when completed, contains a string
describing the state of the EC2 instance
     */
    public CompletableFuture<String> describeEC2InstancesAsync(String newInstanceId)
{
        DescribeInstancesRequest request = DescribeInstancesRequest.builder()
            .instanceIds(newInstanceId)
            .build();

        DescribeInstancesPublisher paginator =
getAsyncClient().describeInstancesPaginator(request);
        AtomicReference<String> publicIpAddressRef = new AtomicReference<>();
        return paginator.subscribe(response -> {
            response.reservations().stream()
                .flatMap(reservation -> reservation.instances().stream())
                .filter(instance -> instance.instanceId().equals(newInstanceId))
                .findFirst()
                .ifPresent(instance ->
publicIpAddressRef.set(instance.publicIpAddress()));
        }).thenApply(v -> {
            String publicIpAddress = publicIpAddressRef.get();
            if (publicIpAddress == null) {
                throw new RuntimeException("Instance with ID " + newInstanceId + "
not found.");
            }
            return publicIpAddress;
        }).exceptionally(ex -> {
            logger.info("Failed to describe instances: " + ex.getMessage());
            throw new RuntimeException("Failed to describe instances", ex);
```

```
        });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeInstances中的。

`DescribeKeyPairs`

以下代码示例演示了如何使用 `DescribeKeyPairs`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Asynchronously describes the key pairs associated with the current AWS
account.
     *
     * @return a {@link CompletableFuture} containing the {@link
DescribeKeyPairsResponse} object, which provides
     * information about the key pairs.
     */
    public CompletableFuture<DescribeKeyPairsResponse> describeKeysAsync() {
        CompletableFuture<DescribeKeyPairsResponse> responseFuture =
getAsyncClient().describeKeyPairs();
        responseFuture.whenComplete((response, exception) -> {
            if (exception != null) {
              throw new RuntimeException("Failed to describe key pairs: " +
exception.getMessage(), exception);
            }
        });

        return responseFuture;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeKeyPairs中的。

DescribeSecurityGroups

以下代码示例演示了如何使用 DescribeSecurityGroups。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Asynchronously describes the security groups for the specified group ID.
    *
    * @param groupName the name of the security group to describe
    * @return a {@link CompletableFuture} that represents the asynchronous
operation
    *         of describing the security groups. The future will complete with a
    *         {@link DescribeSecurityGroupsResponse} object that contains the
    *         security group information.
    */
   public CompletableFuture<String> describeSecurityGroupArnByNameAsync(String
groupName) {
       DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
           .groupNames(groupName)
           .build();

       DescribeSecurityGroupsPublisher paginator =
getAsyncClient().describeSecurityGroupsPaginator(request);
       AtomicReference<String> groupIdRef = new AtomicReference<>();
       return paginator.subscribe(response -> {
           response.securityGroups().stream()
               .filter(securityGroup ->
securityGroup.groupName().equals(groupName))
               .findFirst()
               .ifPresent(securityGroup ->
groupIdRef.set(securityGroup.groupId()));
       }).thenApply(v -> {
           String groupId = groupIdRef.get();
           if (groupId == null) {
```

```
                throw new RuntimeException("No security group found with the name: "
+ groupName);
            }
            return groupId;
        }).exceptionally(ex -> {
            logger.info("Failed to describe security group: " + ex.getMessage());
            throw new RuntimeException("Failed to describe security group", ex);
        });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeSecurityGroups中的。

`DisassociateAddress`

以下代码示例演示了如何使用 `DisassociateAddress`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Disassociates an Elastic IP address from an instance asynchronously.
     *
     * @param associationId The ID of the association you want to disassociate.
     * @return a {@link CompletableFuture} representing the asynchronous operation
of disassociating the address. The
     *         {@link CompletableFuture} will complete with a {@link
DisassociateAddressResponse} when the operation is
     *         finished.
     * @throws RuntimeException if the disassociation of the address fails.
     */
    public CompletableFuture<DisassociateAddressResponse>
disassociateAddressAsync(String associationId) {
        Ec2AsyncClient ec2 = getAsyncClient();
        DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
            .associationId(associationId)
```

```
            .build();

        // Disassociate the address asynchronously.
        CompletableFuture<DisassociateAddressResponse> response =
 ec2.disassociateAddress(addressRequest);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
               throw new RuntimeException("Failed to disassociate address", ex);
            }
        });

        return response;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DisassociateAddress中的。

`GetPasswordData`

以下代码示例演示了如何使用 `GetPasswordData`。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.*;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class GetPasswordData {

   public static void main(String[] args) {
       final String usage = """

               Usage:
                  <instanceId>

               Where:
                  instanceId - An instance id value that you can obtain from the
 AWS Management Console.\s
            """;

       if (args.length != 1) {
           System.out.println(usage);
           System.exit(1);
       }
       String instanceId = args[0];
       Ec2AsyncClient ec2AsyncClient = Ec2AsyncClient.builder()
           .region(Region.US_EAST_1)
           .build();

       try {
           CompletableFuture<Void> future = getPasswordDataAsync(ec2AsyncClient,
 instanceId);
           future.join();
       } catch (RuntimeException rte) {
           System.err.println("An exception occurred: " + (rte.getCause() != null ?
 rte.getCause().getMessage() : rte.getMessage()));
       }
   }

   /**
    * Fetches the password data for the specified EC2 instance asynchronously.
    *
    * @param ec2AsyncClient the EC2 asynchronous client to use for the request
    * @param instanceId instanceId the ID of the EC2 instance for which you want to
 fetch the password data
    * @return a {@link CompletableFuture} that completes when the password data has
 been fetched
    * @throws RuntimeException if there was a failure in fetching the password data
    */
   public static CompletableFuture<Void> getPasswordDataAsync(Ec2AsyncClient
 ec2AsyncClient, String instanceId) {
```

```
        GetPasswordDataRequest getPasswordDataRequest =
 GetPasswordDataRequest.builder()
            .instanceId(instanceId)
            .build();


        CompletableFuture<GetPasswordDataResponse> response =
 ec2AsyncClient.getPasswordData(getPasswordDataRequest);
        response.whenComplete((getPasswordDataResponse, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to get password data for
 instance: " + instanceId, ex);
            } else if (getPasswordDataResponse == null ||
 getPasswordDataResponse.passwordData().isEmpty()) {
                throw new RuntimeException("No password data found for instance: " +
 instanceId);
            } else {
                String encryptedPasswordData =
 getPasswordDataResponse.passwordData();
                System.out.println("Encrypted Password Data: " +
 encryptedPasswordData);
            }
        });

        return response.thenApply(resp -> null);
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetPasswordData中的。

`ReleaseAddress`

以下代码示例演示了如何使用 `ReleaseAddress`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Releases an Elastic IP address asynchronously.
     *
     * @param allocId the allocation ID of the Elastic IP address to be released
     * @return a {@link CompletableFuture} representing the asynchronous operation
of releasing the Elastic IP address
     */
    public CompletableFuture<ReleaseAddressResponse> releaseEC2AddressAsync(String
allocId) {
        ReleaseAddressRequest request = ReleaseAddressRequest.builder()
            .allocationId(allocId)
            .build();

        CompletableFuture<ReleaseAddressResponse> response =
getAsyncClient().releaseAddress(request);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to release Elastic IP address",
ex);
            }
        });

        return response;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ReleaseAddress中的。

`RunInstances`

以下代码示例演示了如何使用 `RunInstances`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
```

```
    * Runs an EC2 instance asynchronously.
    *
    * @param instanceType The instance type to use for the EC2 instance.
    * @param keyName The name of the key pair to associate with the EC2 instance.
    * @param groupName The name of the security group to associate with the EC2
instance.
    * @param amiId The ID of the Amazon Machine Image (AMI) to use for the EC2
instance.
    * @return A {@link CompletableFuture} that completes with the ID of the started
EC2 instance.
    * @throws RuntimeException If there is an error running the EC2 instance.
    */
   public CompletableFuture<String> runInstanceAsync(String instanceType, String
keyName, String groupName, String amiId) {
       RunInstancesRequest runRequest = RunInstancesRequest.builder()
           .instanceType(instanceType)
           .keyName(keyName)
           .securityGroups(groupName)
           .maxCount(1)
           .minCount(1)
           .imageId(amiId)
           .build();

       CompletableFuture<RunInstancesResponse> responseFuture =
getAsyncClient().runInstances(runRequest);
       return responseFuture.thenCompose(response -> {
           String instanceIdVal = response.instances().get(0).instanceId();
           System.out.println("Going to start an EC2 instance and use a waiter to
wait for it to be in running state");
           return getAsyncClient().waiter()
               .waitUntilInstanceExists(r -> r.instanceIds(instanceIdVal))
               .thenCompose(waitResponse -> getAsyncClient().waiter()
                   .waitUntilInstanceRunning(r -> r.instanceIds(instanceIdVal))
                   .thenApply(runningResponse -> instanceIdVal));
       }).exceptionally(throwable -> {
           // Handle any exceptions that occurred during the async call
           throw new RuntimeException("Failed to run EC2 instance: " +
throwable.getMessage(), throwable);
       });
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考RunInstances中的。

StartInstances

以下代码示例演示了如何使用 StartInstances。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Starts an Amazon EC2 instance asynchronously and waits until it is in the
"running" state.
     *
     * @param instanceId the ID of the instance to start
     * @return a {@link CompletableFuture} that completes when the instance has been
started and is in the "running" state, or exceptionally if an error occurs
     */
    public CompletableFuture<Void> startInstanceAsync(String instanceId) {
        StartInstancesRequest startRequest = StartInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
            .client(getAsyncClient())
            .build();

        DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        logger.info("Starting instance " + instanceId + " and waiting for it to
run.");
        CompletableFuture<Void> resultFuture = new CompletableFuture<>();
        return getAsyncClient().startInstances(startRequest)
            .thenCompose(response ->
                ec2Waiter.waitUntilInstanceRunning(describeRequest)
            )
            .thenAccept(waiterResponse -> {
                logger.info("Successfully started instance " + instanceId);
```

```
                resultFuture.complete(null);
            })
            .exceptionally(throwable -> {
                resultFuture.completeExceptionally(new RuntimeException("Failed to
start instance: " + throwable.getMessage(), throwable));
                return null;
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartInstances中的。

`StopInstances`

以下代码示例演示了如何使用 `StopInstances`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Stops the EC2 instance with the specified ID asynchronously and waits for the
instance to stop.
     *
     * @param instanceId the ID of the EC2 instance to stop
     * @return a {@link CompletableFuture} that completes when the instance has been
stopped, or exceptionally if an error occurs
     */
    public CompletableFuture<Void> stopInstanceAsync(String instanceId) {
        StopInstancesRequest stopRequest = StopInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();
```

```
        Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
            .client(getAsyncClient())
            .build();

        CompletableFuture<Void> resultFuture = new CompletableFuture<>();
        logger.info("Stopping instance " + instanceId + " and waiting for it to
stop.");
        getAsyncClient().stopInstances(stopRequest)
            .thenCompose(response -> {
                if (response.stoppingInstances().isEmpty()) {
                    return CompletableFuture.failedFuture(new RuntimeException("No
instances were stopped. Please check the instance ID: " + instanceId));
                }
                return ec2Waiter.waitUntilInstanceStopped(describeRequest);
            })
            .thenAccept(waiterResponse -> {
                logger.info("Successfully stopped instance " + instanceId);
                resultFuture.complete(null);
            })
            .exceptionally(throwable -> {
                logger.error("Failed to stop instance " + instanceId + ": " +
throwable.getMessage(), throwable);
                resultFuture.completeExceptionally(new RuntimeException("Failed to
stop instance: " + throwable.getMessage(), throwable));
                return null;
            });

        return resultFuture;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StopInstances中的。

`TerminateInstances`

以下代码示例演示了如何使用 `TerminateInstances`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Terminates an EC2 instance asynchronously and waits for it to reach the
terminated state.
    *
    * @param instanceId the ID of the EC2 instance to terminate
    * @return a {@link CompletableFuture} that completes when the instance has been
terminated
    * @throws RuntimeException if there is no response from the AWS SDK or if there
is a failure during the termination process
    */
   public CompletableFuture<Object> terminateEC2Async(String instanceId) {
       TerminateInstancesRequest terminateRequest =
TerminateInstancesRequest.builder()
           .instanceIds(instanceId)
           .build();

       CompletableFuture<TerminateInstancesResponse> responseFuture =
getAsyncClient().terminateInstances(terminateRequest);
       return responseFuture.thenCompose(terminateResponse -> {
           if (terminateResponse == null) {
               throw new RuntimeException("No response received for terminating
instance " + instanceId);
           }
           System.out.println("Going to terminate an EC2 instance and use a waiter
to wait for it to be in terminated state");
           return getAsyncClient().waiter()
               .waitUntilInstanceTerminated(r -> r.instanceIds(instanceId))
               .thenApply(waiterResponse -> null);
       }).exceptionally(throwable -> {
           // Handle any exceptions that occurred during the async call
           throw new RuntimeException("Failed to terminate EC2 instance: " +
throwable.getMessage(), throwable);
       });
   }
```

- 有关 API 的详细信息，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 (Amazon EC2) 实例，并将实例数量保持在指定范围内。
- 使用弹性负载均衡处理和分发 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。

适用于 Java 的 SDK 2.x

> **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 wasSuccessul = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
```

```
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group for
your default VPC must
                            allow access from this computer. You can either add it
automatically from this
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            System.out.println("Your load balancer is ready. You can access it by
browsing to:");
            System.out.println("\t http://" + elbDnsName);
        } else {
            System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
            System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
            System.out.println("you can successfully make a GET request to the load
balancer.");
        }
```

```
        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
                """
                        This part of the demonstration shows how to toggle
different parts of the system
                        to create situations where the web service fails, and shows
how using a resilient
                        architecture can keep the web service running in spite of
these failures.

                        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                        The table name is contained in a Systems Manager parameter
named self.param_helper.table.
                        To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
                       """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println(
                """
                        \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
```

```
                        healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
                        """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        Instead of failing when the recommendation service fails,
the web service can return a static response.
                        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
                        """);
        paramHelper.put(paramHelper.failureResponse, "static");

        System.out.println("""
                Now, sending a GET request to the load balancer endpoint returns a
static response.
                The service still reports as healthy because health checks are still
shallow.
                """);
        demoChoices(loadBalancer);

        System.out.println("Let's reinstate the recommendation service.");
        paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

        System.out.println("""
                Let's also substitute bad credentials for one of the instances in
the target group so that it can't
                access the DynamoDB recommendation table. We will get an instance id
value.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        AutoScaler autoScaler = new AutoScaler();

        // Create a new instance profile based on badCredsProfileName.
        templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
        String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
        System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

        String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is " + profileAssociationId);
```

```
        System.out.println("Replacing the profile for instance " + badInstanceId
                + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

        System.out.println(
                """
                        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
                        depending on which instance is selected by the load
balancer.
                        """);

        demoChoices(loadBalancer);

        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
check tests whether
                the web service can access the DynamoDB table that it depends on for
recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
Scaling instance health.
                This kind of deep health check is not recommended for Auto Scaling
instance health, because it
                risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
                """);

        System.out.println("""
                By implementing deep health checks, the load balancer can detect
when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
                Now, checking target health indicates that the instance with bad
credentials
                is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
                instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
```

```
                """);

        demoChoices(loadBalancer);

        System.out.println(
                """
                        Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
                        instance is to terminate it and let the auto scaler start a
new instance to replace it.
                        """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
                Even while the instance is terminating and the new instance is
starting, sending a GET
                request to the web service continues to get a successful
recommendation response because
                the load balancer routes requests to the healthy instances. After
the replacement instance
                starts and reports as healthy, it is included in the load balancing
rotation.
                Note that terminating and replacing an instance typically takes
several minutes, during which time you
                can see the changing health check status until the new instance is
running and healthy.
                """);

        demoChoices(loadBalancer);
        System.out.println(
                "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        String[] actions = {
                "Send a GET request to the load balancer endpoint.",
                "Check the health of load balancer targets.",
                "Go to the next part of the demo."
```

```
        };
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
                System.out.println("-".repeat(88));

                switch (choice) {
                    case 0 -> {
                        System.out.println("Request:\n");
                        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                        CloseableHttpClient httpClient =
HttpClients.createDefault();

                        // Create an HTTP GET request to the ELB.
                        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                        // Execute the request and get the response.
                        HttpResponse response = httpClient.execute(httpGet);
                        int statusCode = response.getStatusLine().getStatusCode();
                        System.out.println("HTTP Status Code: " + statusCode);

                        // Display the JSON response
                        BufferedReader reader = new BufferedReader(
                                new
InputStreamReader(response.getEntity().getContent()));
                        StringBuilder jsonResponse = new StringBuilder();
                        String line;
                        while ((line = reader.readLine()) != null) {
                            jsonResponse.append(line);
                        }
                        reader.close();
```

```
                        // Print the formatted JSON response.
                        System.out.println("Full Response:\n");
                        System.out.println(jsonResponse.toString());

                        // Close the HTTP client.
                        httpClient.close();

                    }
                    case 1 -> {
                        System.out.println("\nChecking the health of load balancer
targets:\n");
                        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
                        for (TargetHealthDescription target : health) {
                            System.out.printf("\tTarget %s on port %d is %s%n",
target.target().id(),
                                    target.target().port(),
target.targetHealth().stateAsString());
                        }
                        System.out.println("""
                                Note that it can take a minute or two for the health
check to update
                                after changes are made.
                                """);
                    }
                    case 2 -> {
                        System.out.println("\nOkay, let's move on.");
                        System.out.println("-".repeat(88));
                        return; // Exit the method when choice is 2
                    }
                    default -> System.out.println("You must choose a value between
0-2. Please select again.");
                }

            } catch (java.util.InputMismatchException e) {
                System.out.println("Invalid input. Please select again.");
                scanner.nextLine(); // Clear the input buffer.
            }
        }
    }

    public static String readFileAsString(String filePath) throws IOException {
        byte[] bytes = Files.readAllBytes(Paths.get(filePath));
        return new String(bytes);
```

```
    }
}
```

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.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
```

```
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
                .instanceIds(instanceId)
                .documentName("AWS-RunShellScript")
                .parameters(Collections.singletonMap("commands",
                        Collections.singletonList("cd / && sudo python3 server.py
80")))
                .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress) {
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(secGroupId)
                .cidrIp(ipAddress)
                .fromPort(Integer.parseInt(port))
                .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                    .builder()
                    .instanceProfileName(profileName)
```

```
                .build();

        GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
        String name = response.instanceProfile().instanceProfileName();
        System.out.println(name);

        RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                .instanceProfileName(profileName)
                .roleName(roleName)
                .build();

        getIAMClient().removeRoleFromInstanceProfile(profileRequest);
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                .instanceProfileName(profileName)
                .build();

        getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
        System.out.println("Deleted instance profile " + profileName);

        DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                .roleName(roleName)
                .build();

        // List attached role policies.
        ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                .listAttachedRolePolicies(role -> role.roleName(roleName));
        List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
        for (AttachedPolicy attachedPolicy : attachedPolicies) {
            DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(attachedPolicy.policyArn())
                    .build();

            getIAMClient().detachRolePolicy(request);
            System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
        }

        getIAMClient().deleteRole(deleteRoleRequest);
        System.out.println("Instance profile and role deleted.");
```

```
        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();


getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network, you
     * must instead specify a prefix list ID. You can also temporarily open the port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     *
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                    .name("group-name")
                    .values("default")
```

```
                    .build();

            Filter filter1 = Filter.builder()
                    .name("vpc-id")
                    .values(VPC)
                    .build();

            DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
                    .filters(filter, filter1)
                    .build();

            DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
                    .describeSecurityGroups(securityGroupsRequest);
            String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
            groupInfo.setGroupName(securityGroup);

            for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
                System.out.println("Found security group: " + secGroup.groupId());

                for (IpPermission ipPermission : secGroup.ipPermissions()) {
                    if (ipPermission.fromPort() == port) {
                        System.out.println("Found inbound rule: " + ipPermission);
                        for (IpRange ipRange : ipPermission.ipRanges()) {
                            String cidrIp = ipRange.cidrIp();
                            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                                System.out.println(cidrIp + " is applicable");
                                portIsOpen = true;
                            }
                        }

                        if (!ipPermission.prefixListIds().isEmpty()) {
                            System.out.println("Prefix lList is applicable");
                            portIsOpen = true;
                        }

                        if (!portIsOpen) {
                            System.out
                                    .println("The inbound rule does not appear to be
open to either this computer's IP,"
                                            + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
```

```
                        } else {
                            break;
                        }
                    }
                }
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }

        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }

    /*
     * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
     * Scaling group.
     * The target group specifies how the load balancer forward requests to the
     * instances
     * in the group.
     */
    public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                    .autoScalingGroupName(asGroupName)
                    .targetGroupARNs(targetGroupARN)
                    .build();


getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {
```

```
        // Get availability zones.
        software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
                .launchTemplate(specification)
                .availabilityZones(zones)
                .maxSize(groupSize)
                .minSize(groupSize)
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }
```

```
    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();

        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
                .builder()
                .filters(defaultFilter)
                .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
                .name("vpc-id")
                .values(vpcId)
                .build();

        Filter azFilter = Filter.builder()
                .name("availability-zone")
                .values(availabilityZones)
                .build();

        Filter defaultForAZ = Filter.builder()
                .name("default-for-az")
                .values("true")
                .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
                .filters(vpcFilter, azFilter, defaultForAZ)
                .build();

        DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }
```

```
   // Gets data about the instances in the EC2 Auto Scaling group.
   public String getBadInstance(String groupName) {
       DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
               .autoScalingGroupNames(groupName)
               .build();

       DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
       AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
       List<String> instanceIds = autoScalingGroup.instances().stream()
               .map(instance -> instance.instanceId())
               .collect(Collectors.toList());

       String[] instanceIdArray = instanceIds.toArray(new String[0]);
       for (String instanceId : instanceIdArray) {
           System.out.println("Instance ID: " + instanceId);
           return instanceId;
       }
       return "";
   }

   // Gets data about the profile associated with an instance.
   public String getInstanceProfile(String instanceId) {
       Filter filter = Filter.builder()
               .name("instance-id")
               .values(instanceId)
               .build();

       DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
               .builder()
               .filters(filter)
               .build();

       DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
               .describeIamInstanceProfileAssociations(associationsRequest);
       return response.iamInstanceProfileAssociations().get(0).associationId();
   }

   public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
```

```
        ListPoliciesRequest listPoliciesRequest =
 ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
 getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to the
 policy.

 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
 listEntitiesRequest =
 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                        .builder()
                        .policyArn(policy.arn())
                        .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                        .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                        || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
 DetachRolePolicyRequest.builder()
                            .policyArn(policy.arn())
                            .roleName(roleName) // Specify the name of the IAM role
                            .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
 DeletePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .build();

                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
                break;
            }
        }

        // List the roles associated with the instance profile
```

```
        ListInstanceProfilesForRoleRequest listRolesRequest =
 ListInstanceProfilesForRoleRequest.builder()
                .roleName(roleName)
                .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
 iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
 listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
 RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(InstanceProfile)
                    .roleName(roleName) // Remove the extra dot here
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance profile
 " + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
 DeleteInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .build();

        getIAMClient().deleteInstanceProfile(r ->
 r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

创建一个包装 ELB 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)
                    .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《AWS SDK for Java 2.x API Reference》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

使用适用于 Java 的 SDK 2.x 的 Amazon ECR 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 ECR 配合使用来执行操作和实现常见场景。

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 基本功能
- 操作

开始使用

开始使用 Amazon ECR

以下代码示例显示了如何开始使用 Amazon ECR。

适用于 Java 的 SDK 2.x

> **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。

```
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 Reference》中的 listImages。

基本功能

了解基本功能

以下代码示例展示了如何：

- 创建 Amazon ECR 存储库。
- 设置存储库策略。
- 检索存储库 URIs。
- 获取 Amazon ECR 授权令牌。
- 设置 Amazon ECR 存储库的生命周期策略。
- 将 Docker 映像推送到 Amazon ECR 存储库。
- 验证 Amazon ECR 存储库中是否存在映像。
- 列出您账户的 Amazon ECR 存储库，并获取有关它们的详细信息。
- 删除 Amazon ECR 存储库。

适用于 Java 的 SDK 2.x

> **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。

运行一个交互式场景，演示 Amazon ECR 的功能。

```
import software.amazon.awssdk.services.ecr.model.EcrException;
import software.amazon.awssdk.services.ecr.model.RepositoryPolicyNotFoundException;

import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code example requires an IAM Role that has permissions to interact with
 the Amazon ECR service.
 *
 * To create an IAM role, see:
 *
 * https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_create.html
 *
 * This Java scenario example requires a local docker image named echo-text. Without
 a local image,
 * this Java program will not successfully run. For more information including how
 to create the local
 * image, see:
 *
 * /scenarios/basics/ecr/README
 *
 */
public class ECRScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    public static void main(String[] args) {
        final String usage = """
```

```
            Usage: <iamRoleARN> <accountId>

            Where:
               iamRoleARN - The IAM role ARN that has the necessary permissions to
access and manage the Amazon ECR repository.
               accountId - Your AWS account number.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            return;
        }

        ECRActions ecrActions = new ECRActions();
        String iamRole = args[0];
        String accountId = args[1];
        String localImageName;

        Scanner scanner = new Scanner(System.in);
        System.out.println("""
             The Amazon Elastic Container Registry (ECR) is a fully-managed Docker
container registry
             service provided by AWS. It allows developers and organizations to
securely
             store, manage, and deploy Docker container images.
             ECR provides a simple and scalable way to manage container images
throughout their lifecycle,
             from building and testing to production deployment.\s

             The `EcrAsyncClient` interface in the AWS SDK for Java 2.x provides a
set of methods to
             programmatically interact with the Amazon ECR service. This allows
developers to
             automate the storage, retrieval, and management of container images as
part of their application
             deployment pipelines. With ECR, teams can focus on building and
deploying their
             applications without having to worry about the underlying
infrastructure required to
             host and manage a container registry.

            This scenario walks you through how to perform key operations for this
service.
            Let's get started...
```

```
            You have two choices:
            1 - Run the entire program.
            2 - Delete an existing Amazon ECR repository named echo-text (created
 from a previous execution of
            this program that did not complete).
            """);

        while (true) {
            String input = scanner.nextLine();
            if (input.trim().equalsIgnoreCase("1")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else if (input.trim().equalsIgnoreCase("2")) {
                String repoName = "echo-text";
                ecrActions.deleteECRRepository(repoName);
                return;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }

        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("""
           1. Create an ECR repository.

           The first task is to ensure we have a local Docker image named echo-
text.
           If this image exists, then an Amazon ECR repository is created.

           An ECR repository is a private Docker container repository provided
           by Amazon Web Services (AWS). It is a managed service that makes it easy
           to store, manage, and deploy Docker container images.\s
           """ );

        // Ensure that a local docker image named echo-text exists.
        boolean doesExist = ecrActions.isEchoTextImagePresent();
        String repoName;
        if (!doesExist){
            System.out.println("The local image named echo-text does not exist");
```

```
            return;
        } else {
            localImageName = "echo-text";
            repoName = "echo-text";
        }

        try {
            String repoArn = ecrActions.createECRRepository(repoName);
            System.out.println("The ARN of the ECR repository is " + repoArn);

        } catch (IllegalArgumentException e) {
            System.err.println("Invalid repository name: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while creating the ECR repository:
" + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        2. Set an ECR repository policy.

        Setting an ECR repository policy using the `setRepositoryPolicy` function is
crucial for maintaining
        the security and integrity of your container images. The repository policy
allows you to
        define specific rules and restrictions for accessing and managing the images
stored within your ECR
        repository.
        """);
        waitForInputToContinue(scanner);
        try {
            ecrActions.setRepoPolicy(repoName, iamRole);

        } catch (RepositoryPolicyNotFoundException e) {
            System.err.println("Invalid repository name: " + e.getMessage());
            return;
        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
```

```
            System.err.println("An error occurred while creating the ECR repository:
" + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        3. Display ECR repository policy.

        Now we will retrieve the ECR policy to ensure it was successfully set.
        """);
        waitForInputToContinue(scanner);
        try {
            String policyText = ecrActions.getRepoPolicy(repoName);
            System.out.println("Policy Text:");
            System.out.println(policyText);

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while creating the ECR repository:
" + e.getMessage());
            return;
        }

        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        4. Retrieve an ECR authorization token.

        You need an authorization token to securely access and interact with the
Amazon ECR registry.
        The `getAuthorizationToken` method of the `EcrAsyncClient` is responsible
for securely accessing
        and interacting with an Amazon ECR repository. This operation is responsible
for obtaining a
        valid authorization token, which is required to authenticate your requests
to the ECR service.

        Without a valid authorization token, you would not be able to perform any
operations on the
```

```
        ECR repository, such as pushing, pulling, or managing your Docker images.

        """);
        waitForInputToContinue(scanner);
        try {
             ecrActions.getAuthToken();

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred while retrieving the authorization
token: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        5. Get the ECR Repository URI.

        The URI  of an Amazon ECR repository is important. When you want to deploy a
container image to
        a container orchestration platform like Amazon Elastic Kubernetes Service
(EKS)
        or Amazon Elastic Container Service (ECS), you need to specify the full
image URI,
        which includes the ECR repository URI. This allows the container runtime to
pull the
        correct container image from the ECR repository.
       """);
        waitForInputToContinue(scanner);

        try {
            ecrActions.getRepositoryURI(repoName);

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;

        } catch (RuntimeException e) {
            System.err.println("An error occurred while retrieving the URI: " +
e.getMessage());
            return;
```

```
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
            6. Set an ECR Lifecycle Policy.

            An ECR Lifecycle Policy is used to manage the lifecycle of Docker images
stored in your ECR repositories.
            These policies allow you to automatically remove old or unused Docker
images from your repositories,
            freeing up storage space and reducing costs.

            This example policy helps to maintain the size and efficiency of the
container registry
            by automatically removing older and potentially unused images, ensuring
that the
            storage is optimized and the registry remains up-to-date.
            """);
        waitForInputToContinue(scanner);
        try {
            ecrActions.setLifeCyclePolicy(repoName);

        } catch (RuntimeException e) {
            System.err.println("An error occurred while setting the lifecycle
policy: " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("""
        7. Push a docker image to the Amazon ECR Repository.

        The `pushImageCmd()` method pushes a local Docker image to an Amazon ECR
repository.
        It sets up the Docker client by connecting to the local Docker host using
the default port.
        It then retrieves the authorization token for the ECR repository by making a
call to the AWS SDK.

        The method uses the authorization token to create an `AuthConfig` object,
which is used to authenticate
```

```
        the Docker client when pushing the image. Finally, the method tags the
Docker image with the specified
        repository name and image tag, and then pushes the image to the ECR
repository using the Docker client.
        If the push operation is successful, the method prints a message indicating
that the image was pushed to ECR.
        """);
        waitForInputToContinue(scanner);

        try {
            ecrActions.pushDockerImage(repoName, localImageName);

        } catch (RuntimeException e) {
            System.err.println("An error occurred while pushing a local Docker image
to Amazon ECR: " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("8. Verify if the image is in the ECR Repository.");
        waitForInputToContinue(scanner);
        try {
            ecrActions.verifyImage(repoName, localImageName);

        } catch (EcrException e) {
            System.err.println("An ECR exception occurred: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("An error occurred " + e.getMessage());
            e.printStackTrace();
            return;
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("9. As an optional step, you can interact with the image
in Amazon ECR by using the CLI.");
        System.out.println("Would you like to view instructions on how to use the
CLI to run the image? (y/n)");
        String ans = scanner.nextLine().trim();
        if (ans.equalsIgnoreCase("y")) {
            String instructions = """
```

```
            1. Authenticate with ECR - Before you can pull the image from Amazon
 ECR, you need to authenticate with the registry. You can do this using the AWS CLI:

                aws ecr get-login-password --region us-east-1 | docker login --
username AWS --password-stdin %s.dkr.ecr.us-east-1.amazonaws.com

            2. Describe the image using this command:

               aws ecr describe-images --repository-name %s --image-ids imageTag=%s

            3. Run the Docker container and view the output using this command:

               docker run --rm %s.dkr.ecr.us-east-1.amazonaws.com/%s:%s
            """;

            instructions = String.format(instructions, accountId, repoName,
 localImageName, accountId, repoName, localImageName);
            System.out.println(instructions);
        }
        waitForInputToContinue(scanner);

        System.out.println(DASHES);
        System.out.println("10. Delete the ECR Repository.");
        System.out.println(
        """
        If the repository isn't empty, you must either delete the contents of the
 repository
        or use the force option (used in this scenario) to delete the repository and
 have Amazon ECR delete all of its contents
        on your behalf.
        """);
        System.out.println("Would you like to delete the Amazon ECR Repository? (y/
n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            System.out.println("You selected to delete the AWS ECR resources.");

            try {
                ecrActions.deleteECRRepository(repoName);

            } catch (EcrException e) {
                System.err.println("An ECR exception occurred: " + e.getMessage());
                return;
            } catch (RuntimeException e) {
```

```
                System.err.println("An error occurred while deleting the Docker
 image: " + e.getMessage());
                e.printStackTrace();
                return;
            }
        }

        System.out.println(DASHES);
        System.out.println("This concludes the Amazon ECR SDK scenario");
        System.out.println(DASHES);
    }

   private static void waitForInputToContinue(Scanner scanner) {
       while (true) {
           System.out.println("");
           System.out.println("Enter 'c' followed by <ENTER> to continue:");
           String input = scanner.nextLine();

           if (input.trim().equalsIgnoreCase("c")) {
               System.out.println("Continuing with the program...");
               System.out.println("");
               break;
           } else {
               // Handle invalid input.
               System.out.println("Invalid input. Please try again.");
           }
       }
   }
}
```

Amazon ECR SDK 方法的封装类。

```
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《AWS SDK for Java 2.x API Reference》中的以下主题。
  - CreateRepository
  - DeleteRepository
  - DescribeImages
  - DescribeRepositories
  - GetAuthorizationToken
  - GetRepositoryPolicy
  - SetRepositoryPolicy
  - StartLifecyclePolicyPreview

操作

`CreateRepository`

以下代码示例演示了如何使用 `CreateRepository`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateRepository中的。

`DeleteRepository`

以下代码示例演示了如何使用 `DeleteRepository`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteRepository中的。

`DescribeImages`

以下代码示例演示了如何使用 `DescribeImages`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeImages中的。

DescribeRepositories

以下代码示例演示了如何使用 DescribeRepositories。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeRepositories中的。

`GetAuthorizationToken`

以下代码示例演示了如何使用 `GetAuthorizationToken`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[GetAuthorizationToken](#)中的。

GetRepositoryPolicy

以下代码示例演示了如何使用 GetRepositoryPolicy。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetRepositoryPolicy中的。

PushImageCmd

以下代码示例演示了如何使用 PushImageCmd。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[PushImageCmd](#)中的。

SetRepositoryPolicy

以下代码示例演示了如何使用 SetRepositoryPolicy。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[SetRepositoryPolicy](中的。

`StartLifecyclePolicyPreview`

以下代码示例演示了如何使用 `StartLifecyclePolicyPreview`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
```

```
    * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartLifecyclePolicyPreview中的。

使用 SDK for Java 2.x 的 Amazon ECS 示例

以下代码示例向您展示了如何在 Amazon ECS 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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从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`

以下代码示例演示了如何使用 `CreateCluster`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateCluster中的。

`CreateService`

以下代码示例演示了如何使用 `CreateService`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateService中的。

`DeleteService`

以下代码示例演示了如何使用 `DeleteService`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteService中的。

DescribeClusters

以下代码示例演示了如何使用 DescribeClusters。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeClusters中的。

DescribeTasks

以下代码示例演示了如何使用 DescribeTasks。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeTasks中的。

ListClusters

以下代码示例演示了如何使用 ListClusters。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListClusters中的。

`UpdateService`

以下代码示例演示了如何使用 `UpdateService`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考UpdateService中的。

ELB-使用适用于 Java 的 SDK 的第 2 版示例 2.x

以下代码示例向您展示了如何使用 AWS SDK for Java 2.x 与 ELB-版本 2 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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 操作
- 场景

开始使用

你好 ELB

以下代码示例显示了如何开始使用 ELB。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[DescribeLoadBalancers](#)中的。

操作

`CreateListener`

以下代码示例演示了如何使用 `CreateListener`。

适用于 Java 的 SDK 2.x

> **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。

```
    /*
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
 String lbName, int port,
            String protocol) {
```

```
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                    .subnets(subnetIdStrings)
                    .name(lbName)
                    .scheme("internet-facing")
                    .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(lbARN)
                    .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest = CreateListenerRequest.builder()
```

```
.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;

        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateListener中的。

`CreateLoadBalancer`

以下代码示例演示了如何使用 `CreateLoadBalancer`。

适用于 Java 的 SDK 2.x

> **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。

```
    /*
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
```

```
   public String createLoadBalancer(List<Subnet> subnetIds, String targetGroupARN,
String lbName, int port,
           String protocol) {
       try {
           List<String> subnetIdStrings = subnetIds.stream()
                   .map(Subnet::subnetId)
                   .collect(Collectors.toList());

           CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                   .subnets(subnetIdStrings)
                   .name(lbName)
                   .scheme("internet-facing")
                   .build();

           // Create and wait for the load balancer to become available.
           CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
           String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

           ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
           DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                   .loadBalancerArns(lbARN)
                   .build();

           System.out.println("Waiting for Load Balancer " + lbName + " to become
available.");
           WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                   .waitUntilLoadBalancerAvailable(request);
           waiterResponse.matched().response().ifPresent(System.out::println);
           System.out.println("Load Balancer " + lbName + " is available.");

           // Get the DNS name (endpoint) of the load balancer.
           String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
           System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

           // Create a listener for the load balance.
           Action action = Action.builder()
                   .targetGroupArn(targetGroupARN)
                   .type("forward")
                   .build();
```

```
            CreateListenerRequest listenerRequest = CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;

        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateLoadBalancer中的。

`CreateTargetGroup`

以下代码示例演示了如何使用 `CreateTargetGroup`。

适用于 Java 的 SDK 2.x

> **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。

```
    /*
     * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[CreateTargetGroup](#)中的。

`DeleteLoadBalancer`

以下代码示例演示了如何使用 `DeleteLoadBalancer`。

适用于 Java 的 SDK 2.x

> **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。

```
   // 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteLoadBalancer中的。

`DeleteTargetGroup`

以下代码示例演示了如何使用 `DeleteTargetGroup`。

适用于 Java 的 SDK 2.x

> **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。

```
    // 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[DeleteTargetGroup](#)中的。

`DescribeTargetHealth`

以下代码示例演示了如何使用 `DescribeTargetHealth`。

适用于 Java 的 SDK 2.x

> **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。

```
    // 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 的详细信息，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 (Amazon EC2) 实例，并将实例数量保持在指定范围内。
- 使用弹性负载均衡处理和分发 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。

适用于 Java 的 SDK 2.x

> **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 wasSuccessul = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
```

```
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group for
your default VPC must
                            allow access from this computer. You can either add it
automatically from this
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            System.out.println("Your load balancer is ready. You can access it by
browsing to:");
            System.out.println("\t http://" + elbDnsName);
        } else {
            System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
            System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
```

```
            System.out.println("you can successfully make a GET request to the load
balancer.");
        }

        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
                """
                        This part of the demonstration shows how to toggle
different parts of the system
                        to create situations where the web service fails, and shows
how using a resilient
                        architecture can keep the web service running in spite of
these failures.

                        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                        The table name is contained in a Systems Manager parameter
named self.param_helper.table.
                        To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
                       """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");
```

```
        System.out.println(
                """
                        \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
                        healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
                        """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        Instead of failing when the recommendation service fails,
the web service can return a static response.
                        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
                        """);
        paramHelper.put(paramHelper.failureResponse, "static");

        System.out.println("""
                Now, sending a GET request to the load balancer endpoint returns a
static response.
                The service still reports as healthy because health checks are still
shallow.
                """);
        demoChoices(loadBalancer);

        System.out.println("Let's reinstate the recommendation service.");
        paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

        System.out.println("""
                Let's also substitute bad credentials for one of the instances in
the target group so that it can't
                access the DynamoDB recommendation table. We will get an instance id
value.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        AutoScaler autoScaler = new AutoScaler();

        // Create a new instance profile based on badCredsProfileName.
        templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
        String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
```

```
        System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

        String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is " + profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId
                + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

        System.out.println(
                """
                        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
                        depending on which instance is selected by the load
balancer.
                        """);

        demoChoices(loadBalancer);

        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
check tests whether
                the web service can access the DynamoDB table that it depends on for
recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
Scaling instance health.
                This kind of deep health check is not recommended for Auto Scaling
instance health, because it
                risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
                """);

        System.out.println("""
                By implementing deep health checks, the load balancer can detect
when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
                Now, checking target health indicates that the instance with bad
credentials
```

```
                is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
                instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
                """);

        demoChoices(loadBalancer);

        System.out.println(
                """
                        Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
                        instance is to terminate it and let the auto scaler start a
new instance to replace it.
                        """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
                Even while the instance is terminating and the new instance is
starting, sending a GET
                request to the web service continues to get a successful
recommendation response because
                the load balancer routes requests to the healthy instances. After
the replacement instance
                starts and reports as healthy, it is included in the load balancing
rotation.
                Note that terminating and replacing an instance typically takes
several minutes, during which time you
                can see the changing health check status until the new instance is
running and healthy.
                """);

        demoChoices(loadBalancer);
        System.out.println(
                "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }
```

```
   public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
       String[] actions = {
               "Send a GET request to the load balancer endpoint.",
               "Check the health of load balancer targets.",
               "Go to the next part of the demo."
       };
       Scanner scanner = new Scanner(System.in);

       while (true) {
           System.out.println("-".repeat(88));
           System.out.println("See the current state of the service by selecting
one of the following choices:");
           for (int i = 0; i < actions.length; i++) {
               System.out.println(i + ": " + actions[i]);
           }

           try {
               System.out.print("\nWhich action would you like to take? ");
               int choice = scanner.nextInt();
               System.out.println("-".repeat(88));

               switch (choice) {
                   case 0 -> {
                       System.out.println("Request:\n");
                       System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                       CloseableHttpClient httpClient =
HttpClients.createDefault();

                       // Create an HTTP GET request to the ELB.
                       HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                       // Execute the request and get the response.
                       HttpResponse response = httpClient.execute(httpGet);
                       int statusCode = response.getStatusLine().getStatusCode();
                       System.out.println("HTTP Status Code: " + statusCode);

                       // Display the JSON response
                       BufferedReader reader = new BufferedReader(
                               new
InputStreamReader(response.getEntity().getContent()));
                       StringBuilder jsonResponse = new StringBuilder();
```

```
                        String line;
                        while ((line = reader.readLine()) != null) {
                            jsonResponse.append(line);
                        }
                        reader.close();

                        // Print the formatted JSON response.
                        System.out.println("Full Response:\n");
                        System.out.println(jsonResponse.toString());

                        // Close the HTTP client.
                        httpClient.close();

                    }
                    case 1 -> {
                        System.out.println("\nChecking the health of load balancer
targets:\n");
                        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
                        for (TargetHealthDescription target : health) {
                            System.out.printf("\tTarget %s on port %d is %s%n",
target.target().id(),
                                    target.target().port(),
target.targetHealth().stateAsString());
                        }
                        System.out.println("""
                                Note that it can take a minute or two for the health
check to update
                                after changes are made.
                                """);
                    }
                    case 2 -> {
                        System.out.println("\nOkay, let's move on.");
                        System.out.println("-".repeat(88));
                        return; // Exit the method when choice is 2
                    }
                    default -> System.out.println("You must choose a value between
0-2. Please select again.");
                }

            } catch (java.util.InputMismatchException e) {
                System.out.println("Invalid input. Please select again.");
                scanner.nextLine(); // Clear the input buffer.
            }
```

```
        }
    }

    public static String readFileAsString(String filePath) throws IOException {
        byte[] bytes = Files.readAllBytes(Paths.get(filePath));
        return new String(bytes);
    }
}
```

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.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
                .instanceIds(instanceId)
                .documentName("AWS-RunShellScript")
                .parameters(Collections.singletonMap("commands",
                        Collections.singletonList("cd / && sudo python3 server.py
80")))
                .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress) {
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(secGroupId)
                .cidrIp(ipAddress)
                .fromPort(Integer.parseInt(port))
                .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
```

```
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                    .builder()
                    .instanceProfileName(profileName)
                    .build();

            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .roleName(roleName)
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .build();

            getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
            System.out.println("Deleted instance profile " + profileName);

            DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            // List attached role policies.
            ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                    .listAttachedRolePolicies(role -> role.roleName(roleName));
            List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
            for (AttachedPolicy attachedPolicy : attachedPolicies) {
                DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(attachedPolicy.policyArn())
                        .build();

                getIAMClient().detachRolePolicy(request);
```

```
                System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
            }

            getIAMClient().deleteRole(deleteRoleRequest);
            System.out.println("Instance profile and role deleted.");

        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();


getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network, you
     * must instead specify a prefix list ID. You can also temporarily open the port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     *
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
```

```
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                    .name("group-name")
                    .values("default")
                    .build();

            Filter filter1 = Filter.builder()
                    .name("vpc-id")
                    .values(VPC)
                    .build();

            DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
                    .filters(filter, filter1)
                    .build();

            DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
                    .describeSecurityGroups(securityGroupsRequest);
            String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
            groupInfo.setGroupName(securityGroup);

            for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
                System.out.println("Found security group: " + secGroup.groupId());

                for (IpPermission ipPermission : secGroup.ipPermissions()) {
                    if (ipPermission.fromPort() == port) {
                        System.out.println("Found inbound rule: " + ipPermission);
                        for (IpRange ipRange : ipPermission.ipRanges()) {
                            String cidrIp = ipRange.cidrIp();
                            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                                System.out.println(cidrIp + " is applicable");
                                portIsOpen = true;
                            }
                        }

                        if (!ipPermission.prefixListIds().isEmpty()) {
                            System.out.println("Prefix lList is applicable");
                            portIsOpen = true;
                        }
```

```
                        if (!portIsOpen) {
                            System.out
                                    .println("The inbound rule does not appear to be
open to either this computer's IP,"
                                            + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
                        } else {
                            break;
                        }
                    }
                }
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }

        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }

    /*
     * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
     * Scaling group.
     * The target group specifies how the load balancer forward requests to the
     * instances
     * in the group.
     */
    public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                    .autoScalingGroupName(asGroupName)
                    .targetGroupARNs(targetGroupARN)
                    .build();


getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

        // Get availability zones.
        software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
                .launchTemplate(specification)
                .availabilityZones(zones)
                .maxSize(groupSize)
                .minSize(groupSize)
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }

    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();

        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
                .builder()
                .filters(defaultFilter)
                .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
                .name("vpc-id")
                .values(vpcId)
                .build();

        Filter azFilter = Filter.builder()
                .name("availability-zone")
                .values(availabilityZones)
                .build();

        Filter defaultForAZ = Filter.builder()
                .name("default-for-az")
                .values("true")
                .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
                .filters(vpcFilter, azFilter, defaultForAZ)
```

```
                .build();

        DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

        DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
                .map(instance -> instance.instanceId())
                .collect(Collectors.toList());

        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }

    // Gets data about the profile associated with an instance.
    public String getInstanceProfile(String instanceId) {
        Filter filter = Filter.builder()
                .name("instance-id")
                .values(instanceId)
                .build();

        DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
                .builder()
                .filters(filter)
                .build();

        DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
                .describeIamInstanceProfileAssociations(associationsRequest);
```

```
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
 InstanceProfile) {
        ListPoliciesRequest listPoliciesRequest =
 ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
 getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to the
 policy.

 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
 listEntitiesRequest =
 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                        .builder()
                        .policyArn(policy.arn())
                        .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                        .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                        || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
 DetachRolePolicyRequest.builder()
                            .policyArn(policy.arn())
                            .roleName(roleName) // Specify the name of the IAM role
                            .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
 DeletePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .build();

                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
```

```
                break;
            }
        }

        // List the roles associated with the instance profile
        ListInstanceProfilesForRoleRequest listRolesRequest =
 ListInstanceProfilesForRoleRequest.builder()
                .roleName(roleName)
                .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
 iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
 listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
 RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(InstanceProfile)
                    .roleName(roleName) // Remove the extra dot here
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance profile
 " + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
 DeleteInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .build();

        getIAMClient().deleteInstanceProfile(r ->
 r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

创建一个包装 ELB 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)
                    .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《AWS SDK for Java 2.x API Reference》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 使用适用于 Java 的 SDK 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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从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`

以下代码示例演示了如何使用 `CreateContainer`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateContainer中的。

DeleteContainer

以下代码示例演示了如何使用 DeleteContainer。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteContainer中的。

`DeleteObject`

以下代码示例演示了如何使用 `DeleteObject`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteObject中的。

`DescribeContainer`

以下代码示例演示了如何使用 `DescribeContainer`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeContainer中的。

`GetObject`

以下代码示例演示了如何使用 `GetObject`。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.core.ResponseInputStream;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediastore.MediaStoreClient;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerRequest;
import software.amazon.awssdk.services.mediastore.model.DescribeContainerResponse;
import software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient;
import software.amazon.awssdk.services.mediastoredata.model.GetObjectRequest;
import software.amazon.awssdk.services.mediastoredata.model.GetObjectResponse;
import software.amazon.awssdk.services.mediastoredata.model.MediaStoreDataException;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import java.net.URI;
import java.net.URISyntaxException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
 */
public class GetObject {
    public static void main(String[] args) throws URISyntaxException {
        final String usage = """

                Usage:    <completePath> <containerName> <savePath>

                Where:
                   completePath - The path of the object in the container (for
 example, Videos5/sampleVideo.mp4).
                   containerName - The name of the container.
                   savePath - The path on the local drive where the file is saved,
 including the file name (for example, C:/AWS/myvid.mp4).
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String completePath = args[0];
        String containerName = args[1];
        String savePath = args[2];

        Region region = Region.US_EAST_1;
        URI uri = new URI(getEndpoint(containerName));
        MediaStoreDataClient mediaStoreData = MediaStoreDataClient.builder()
                .endpointOverride(uri)
                .region(region)
                .build();

        getMediaObject(mediaStoreData, completePath, savePath);
        mediaStoreData.close();
    }

    public static void getMediaObject(MediaStoreDataClient mediaStoreData, String
 completePath, String savePath) {

        try {
            GetObjectRequest objectRequest = GetObjectRequest.builder()
                    .path(completePath)
                    .build();

            // Write out the data to a file.
```

```
            ResponseInputStream<GetObjectResponse> data =
 mediaStoreData.getObject(objectRequest);
            byte[] buffer = new byte[data.available()];
            data.read(buffer);

            File targetFile = new File(savePath);
            OutputStream outStream = new FileOutputStream(targetFile);
            outStream.write(buffer);
            System.out.println("The data was written to " + savePath);

        } catch (MediaStoreDataException | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    private static String getEndpoint(String containerName) {
        Region region = Region.US_EAST_1;
        MediaStoreClient mediaStoreClient = MediaStoreClient.builder()
                .region(region)
                .build();

        DescribeContainerRequest containerRequest =
 DescribeContainerRequest.builder()
                .containerName(containerName)
                .build();

        DescribeContainerResponse response =
 mediaStoreClient.describeContainer(containerRequest);
        return response.container().endpoint();
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetObject中的。

`ListContainers`

以下代码示例演示了如何使用 `ListContainers`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListContainers中的。

`PutObject`

以下代码示例演示了如何使用 `PutObject`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutObject中的。

AWS Entity Resolution 数据匹配服务 使用适用于 Java 的 SDK 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 Entity Resolution 数据匹配服务。 AWS SDK for Java 2.x

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 基本功能
- 操作

开始使用

你好 AWS Entity Resolution 数据匹配服务

以下代码示例展示了如何开始使用 AWS Entity Resolution 数据匹配服务。

适用于 Java 的 SDK 2.x

> **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。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloEntityResoultion {

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

    private static EntityResolutionAsyncClient entityResolutionAsyncClient;
    public static void main(String[] args) {
```

```
        listMatchingWorkflows();
    }

    public static EntityResolutionAsyncClient getResolutionAsyncClient() {
        if (entityResolutionAsyncClient == null) {
            /*
            The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
version 2,
            and it is designed to provide a high-performance, asynchronous HTTP
client for interacting with AWS services.
             It uses the Netty framework to handle the underlying network
communication and the Java NIO API to
             provide a non-blocking, event-driven approach to HTTP requests and
responses.
             */

            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(50)  // Adjust as needed.
                .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
                .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
                .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
timeout.
                .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
individual call attempt timeout.
                .retryStrategy(RetryMode.STANDARD)
                .build();

            entityResolutionAsyncClient = EntityResolutionAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return entityResolutionAsyncClient;
    }

    /**
     * Lists all matching workflows using an asynchronous paginator.
     * <p>
```

```
     * This method requests a paginated list of matching workflows from the
     * AWS Entity Resolution service and logs the names of the retrieved workflows.
     * It uses an asynchronous approach with a paginator and waits for the operation
     * to complete using {@code CompletableFuture#join()}.
     * </p>
     */
    public static void listMatchingWorkflows() {
        ListMatchingWorkflowsRequest request =
 ListMatchingWorkflowsRequest.builder().build();

        ListMatchingWorkflowsPublisher paginator =
            getResolutionAsyncClient().listMatchingWorkflowsPaginator(request);

        // Iterate through the paginated results asynchronously
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            response.workflowSummaries().forEach(workflow ->
                logger.info("Matching Workflow Name: " + workflow.workflowName())
            );
        });

        // Wait for the asynchronous operation to complete
        future.join();
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListMatchingWorkflows中的。

基本功能

了解基本功能

以下代码示例展示了如何：

- 创建架构映射。
- 创建 AWS Entity Resolution 数据匹配服务 工作流程。
- 启动工作流程的匹配作业。
- 获取匹配作业的详细信息。
- 获取架构映射。
- 列出所有架构映射。

- 为架构映射资源添加标签。
- 删除资 AWS Entity Resolution 数据匹配服务 产。

适用于 Java 的 SDK 2.x

> **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。

运行演示 AWS Entity Resolution 数据匹配服务 功能的交互式场景。

```
public class EntityResScenario {
    private static final Logger logger =
 LoggerFactory.getLogger(EntityResScenario.class);
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final String STACK_NAME = "EntityResolutionCdkStack2";
    private static final String ENTITY_RESOLUTION_ROLE_ARN_KEY =
 "EntityResolutionRoleArn";
    private static final String GLUE_DATA_BUCKET_NAME_KEY = "GlueDataBucketName";
    private static final String JSON_GLUE_TABLE_ARN_KEY = "JsonErGlueTableArn";
    private static final String CSV_GLUE_TABLE_ARN_KEY = "CsvErGlueTableArn";
    private static String glueBucketName;
    private static String workflowName = "workflow-" + UUID.randomUUID();

    private static String jsonSchemaMappingName = "jsonschema-" + UUID.randomUUID();
    private static String jsonSchemaMappingArn = null;
    private static String csvSchemaMappingName = "csv-" + UUID.randomUUID();
    private static String roleARN;
    private static String csvGlueTableArn;
    private static String jsonGlueTableArn;
    private static Scanner scanner = new Scanner(System.in);

    private static EntityResActions actions = new EntityResActions();

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

        logger.info("Welcome to the AWS Entity Resolution Scenario.");
        logger.info("""
```

```
            AWS Entity Resolution is a fully-managed machine learning service
provided by
            Amazon Web Services (AWS) that helps organizations extract, link, and
            organize information from multiple data sources. It leverages natural
            language processing and deep learning models to identify and resolve
            entities, such as people, places, organizations, and products,
            across structured and unstructured data.

            With Entity Resolution, customers can build robust data integration
            pipelines to combine and reconcile data from multiple systems,
databases,
            and documents. The service can handle ambiguous, incomplete, or
conflicting
            information, and provide a unified view of entities and their
relationships.
            This can be particularly valuable in applications such as customer 360,
            fraud detection, supply chain management, and knowledge management,
where
            accurate entity identification is crucial.

            The `EntityResolutionAsyncClient` interface in the AWS SDK for Java 2.x
            provides a set of methods to programmatically interact with the AWS
Entity
            Resolution service. This allows developers to automate the entity
extraction,
            linking, and deduplication process as part of their data processing
workflows.
            With Entity Resolution, organizations can unlock the value of their
data,
            improve decision-making, and enhance customer experiences by having a
reliable,
            comprehensive view of their key entities.
            """);

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("""
            To prepare the AWS resources needed for this scenario application, the
next step uploads
            a CloudFormation template whose resulting stack creates the following
resources:
            - An AWS Glue Data Catalog table
```

```
            - An AWS IAM role
            - An AWS S3 bucket
            - An AWS Entity Resolution Schema

            It can take a couple minutes for the Stack to finish creating the
resources.
            """);
        waitForInputToContinue(scanner);
        logger.info("Generating resources...");
        CloudFormationHelper.deployCloudFormationStack(STACK_NAME);
        Map<String, String> outputsMap =
CloudFormationHelper.getStackOutputsAsync(STACK_NAME).join();
        roleARN = outputsMap.get(ENTITY_RESOLUTION_ROLE_ARN_KEY);
        glueBucketName = outputsMap.get(GLUE_DATA_BUCKET_NAME_KEY);
        csvGlueTableArn = outputsMap.get(CSV_GLUE_TABLE_ARN_KEY);
        jsonGlueTableArn = outputsMap.get(JSON_GLUE_TABLE_ARN_KEY);
        logger.info(DASHES);
        waitForInputToContinue(scanner);

        try {
            runScenario();

        } catch (Exception ce) {
            Throwable cause = ce.getCause();
            logger.error("An exception happened: " + (cause != null ?
cause.getMessage() : ce.getMessage()));
        }
    }

    private static void runScenario() throws InterruptedException {
        /*
         This JSON is a valid input for the AWS Entity Resolution service.
         The JSON represents an array of three objects, each containing an "id",
"name", and "email"
         property. This format aligns with the expected input structure for the
         Entity Resolution service.
         */
        String json = """
            {"id":"1","name":"Jane Doe","email":"jane.doe@example.com"}
            {"id":"2","name":"John Doe","email":"john.doe@example.com"}
            {"id":"3","name":"Jorge Souza","email":"jorge_souza@example.com"}
            """;
        logger.info("Upload the following JSON objects to the {} S3 bucket.",
glueBucketName);
```

```
        logger.info(json);
        String csv = """
            id,name,email,phone
            1,Jane B.,Doe,jane.doe@example.com,555-876-9846
            2,John Doe Jr.,john.doe@example.com,555-654-3210
            3,María García,maría_garcia@company.com,555-567-1234
            4,Mary Major,mary_major@company.com,555-222-3333
            """;
        logger.info("Upload the following CSV data to the {} S3 bucket.",
 glueBucketName);
        logger.info(csv);
        waitForInputToContinue(scanner);
        try {
            actions.uploadInputData(glueBucketName, json, csv);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();

            if (cause == null) {
                logger.error("Failed to upload input data: {}", ce.getMessage(),
 ce);
            }

            if (cause instanceof ResourceNotFoundException) {
                logger.error("Failed to upload input data as the resource was not
 found: {}", cause.getMessage(), cause);
            }
            return;
        }
        logger.info("The JSON and CSV objects have been uploaded to the S3
 bucket.");
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("1. Create Schema Mapping");
        logger.info("""
            Entity Resolution schema mapping aligns and integrates data from
            multiple sources by identifying and matching corresponding entities
            like customers or products. It unifies schemas, resolves conflicts,
            and uses machine learning to link related entities, enabling a
            consolidated, accurate view for improved data quality and decision-
making.
```

```
            In this example, the schema mapping lines up with the fields in the JSON
and CSV objects. That is,
            it contains these fields: id, name, and email.
            """);
        try {
            CreateSchemaMappingResponse response =
actions.createSchemaMappingAsync(jsonSchemaMappingName).join();
            jsonSchemaMappingName = response.schemaName();
            logger.info("The JSON schema mapping name is " + jsonSchemaMappingName);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();

            if (cause == null) {
                logger.error("Failed to create JSON schema mapping: {}",
ce.getMessage(), ce);
            }

            if (cause instanceof ConflictException) {
                logger.error("Schema mapping conflict detected: {}",
cause.getMessage(), cause);
            } else {
                logger.error("Unexpected error while creating schema mapping: {}",
cause.getMessage(), cause);
            }
            return;
        }

        try {
            CreateSchemaMappingResponse response =
actions.createSchemaMappingAsync(csvSchemaMappingName).join();
            csvSchemaMappingName = response.schemaName();
            logger.info("The CSV schema mapping name is " + csvSchemaMappingName);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause == null) {
                logger.error("Failed to create CSV schema mapping: {}",
ce.getMessage(), ce);
            }

            if (cause instanceof ConflictException) {
                logger.error("Schema mapping conflict detected: {}",
cause.getMessage(), cause);
            } else {
```

```
                logger.error("Unexpected error while creating CSV schema mapping:
{}", cause.getMessage(), cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Create an AWS Entity Resolution Workflow. ");
        logger.info("""
            An Entity Resolution matching workflow identifies and links records
            across datasets that represent the same real-world entity, such as
            customers or products. Using techniques like schema mapping,
            data profiling, and machine learning algorithms,
            it evaluates attributes like names or emails to detect duplicates
            or relationships, even with variations or inconsistencies.
            The workflow outputs consolidated, de-duplicated data.

            We will use the machine learning-based matching technique.
            """);
        waitForInputToContinue(scanner);
        try {
            String workflowArn = actions.createMatchingWorkflowAsync(
                roleARN, workflowName, glueBucketName, jsonGlueTableArn,
                jsonSchemaMappingName, csvGlueTableArn,
csvSchemaMappingName).join();

            logger.info("The workflow ARN is: " + workflowArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();

            if (cause == null) {
                logger.error("An unexpected error occurred: {}", ce.getMessage(),
ce);
            }

            if (cause instanceof ValidationException) {
                logger.error("Validation error: {}", cause.getMessage(), cause);
            } else if (cause instanceof ConflictException) {
                logger.error("Workflow conflict detected: {}", cause.getMessage(),
cause);
            } else {
                logger.error("Unexpected error: {}", cause.getMessage(), cause);
```

```
            }
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);
        logger.info("3. Start the matching job of the " + workflowName + "
workflow.");
        waitForInputToContinue(scanner);
        String jobId = null;
        try {
            jobId = actions.startMatchingJobAsync(workflowName).join();
            logger.info("The matching job was successfully started.");
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ConflictException) {
                logger.error("Job conflict detected: {}", cause.getMessage(),
cause);
            } else {
                logger.error("Unexpected error while starting the job: {}",
ce.getMessage(), ce);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. While the matching job is running, let's look at other API
methods. First, let's get details for job " + jobId);
        waitForInputToContinue(scanner);
        try {
            actions.getMatchingJobAsync(jobId, workflowName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The matching job not found: {}", cause.getMessage(),
cause);
            } else {
                logger.error("Failed to start matching job: " + (cause != null ?
cause.getMessage() : ce.getMessage()));
            }
            return;
        }
```

```
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Get the schema mapping for the JSON data.");
        waitForInputToContinue(scanner);
        try {
            GetSchemaMappingResponse response =
actions.getSchemaMappingAsync(jsonSchemaMappingName).join();
            jsonSchemaMappingArn = response.schemaArn();
            logger.info("Schema mapping ARN is " + jsonSchemaMappingArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("Schema mapping not found: {}", cause.getMessage(),
cause);
            } else {
                logger.error("Error retrieving the specific schema mapping: " +
ce.getCause().getMessage());
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. List Schema Mappings.");
        try {
            actions.ListSchemaMappings();
        } catch (CompletionException ce) {
            logger.error("Error retrieving schema mappings: " +
ce.getCause().getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Tag the {} resource.", jsonSchemaMappingName);
        logger.info("""
            Tags can help you organize and categorize your Entity Resolution
resources.
            You can also use them to scope user permissions by granting a user
permission
            to access or change only resources with certain tag values.
```

```
            In Entity Resolution, SchemaMapping and MatchingWorkflow can be tagged.
For this example,
            the SchemaMapping is tagged.
                """);
        try {
            actions.tagEntityResource(jsonSchemaMappingArn).join();
        } catch (CompletionException ce) {
            logger.error("Error tagging the resource: " +
ce.getCause().getMessage());
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. View the results of the AWS Entity Resolution Workflow.");
        logger.info("""
            You cannot view the result of the workflow that is in a running state.
            In order to view the results, you need to wait for the workflow that we
started in step 3 to complete.

            If you choose not to wait, you cannot view the results. You can perform

            this task manually in the AWS Management Console.

            This can take up to 30 mins (y/n).
            """);
        String viewAns = scanner.nextLine().trim();
        boolean isComplete = false;
        if (viewAns.equalsIgnoreCase("y")) {
            logger.info("You selected to view the Entity Resolution Workflow
results.");
            countdownWithWorkflowCheck(actions, 1800, jobId, workflowName);
            isComplete = true;
            try {
                JobMetrics metrics = actions.getJobInfo(workflowName, jobId).join();
                logger.info("Number of input records: {}", metrics.inputRecords());
                logger.info("Number of match ids: {}", metrics.matchIDs());
                logger.info("Number of records not processed: {}",
metrics.recordsNotProcessed());
                logger.info("Number of total records processed: {}",
metrics.totalRecordsProcessed());
```

```
                logger.info("The following represents the output data generated by
the Entity Resolution workflow based on the JSON and CSV input data. The output
data is stored in the {} bucket.", glueBucketName);
                actions.printData(glueBucketName);

                logger.info("""

                    Note that each of the last 2 records are considered a match even
though the 'name' differs between the records;
                    For example 'John Doe Jr.' compared to 'John Doe'.
                    The confidence level is a value between 0 and 1, where 1
indicates a perfect match.

                    """);

            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    logger.error("The job not found: {}", cause.getMessage(),
cause);
                } else {
                    logger.error("Error retrieving job information: " +
ce.getCause().getMessage());
                }
                return;
            }
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("9. Do you want to delete the resources, including the workflow?
(y/n)");
        logger.info("""
            You cannot delete the workflow that is in a running state.
            In order to delete the workflow, you need to wait for the workflow to
complete.

            You can delete the workflow manually in the AWS Management Console at a
later time.

            If you already waited for the workflow to complete in the previous
step,
```

```
            the workflow is completed and you can delete it.

            If the workflow is not completed, this can take up to 30 mins (y/n).
            """);
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            try {
                if (!isComplete) {
                    countdownWithWorkflowCheck(actions, 1800, jobId, workflowName);
                }
                actions.deleteMatchingWorkflowAsync(workflowName).join();
                logger.info("Workflow deleted successfully!");
            } catch (CompletionException ce) {
                logger.info("Error deleting the workflow: {} ", ce.getMessage());
                return;
            }

            try {
                // Delete both schema mappings.
                actions.deleteSchemaMappingAsync(jsonSchemaMappingName).join();
                actions.deleteSchemaMappingAsync(csvSchemaMappingName).join();
                logger.info("Both schema mappings were deleted successfully!");
            } catch (CompletionException ce) {
                logger.error("Error deleting schema mapping: {}", ce.getMessage());
                return;
            }

            waitForInputToContinue(scanner);
            logger.info(DASHES);
            logger.info("""
                Now we delete the CloudFormation stack, which deletes
                the resources that were created at the beginning of this scenario.
                """);
            waitForInputToContinue(scanner);
            logger.info(DASHES);
            try {
                deleteCloudFormationStack();
            } catch (RuntimeException e) {
                logger.error("Failed to delete the stack: {}", e.getMessage());
                return;
            }

        } else {
```

```
            logger.info("You can delete the AWS resources in the AWS Management
Console.");
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("This concludes the AWS Entity Resolution scenario.");
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }

    public static void countdownWithWorkflowCheck(EntityResActions actions, int
totalSeconds, String jobId, String workflowName) throws InterruptedException {
        int secondsElapsed = 0;

        while (true) {
            // Calculate display minutes and seconds.
            int remainingTime = totalSeconds - secondsElapsed;
            int displayMinutes = remainingTime / 60;
            int displaySeconds = remainingTime % 60;

            // Print the countdown.
            System.out.printf("\r%02d:%02d", displayMinutes, displaySeconds);
            Thread.sleep(1000); // Wait for 1 second
            secondsElapsed++;
```

```
            // Check workflow status every 60 seconds.
            if (secondsElapsed % 60 == 0 || remainingTime <= 0) {
                GetMatchingJobResponse response =
 actions.checkWorkflowStatusCompleteAsync(jobId, workflowName).join();
                if (response != null &&
 "SUCCEEDED".equalsIgnoreCase(String.valueOf(response.status()))) {
                    logger.info(""); // Move to the next line after countdown.
                    logger.info("Countdown complete: Workflow is in Completed
 state!");
                    break; // Break out of the loop if the status is "SUCCEEDED"
                }
            }

            // If countdown reaches zero, reset it for continuous countdown.
            if (remainingTime <= 0) {
                secondsElapsed = 0;
            }
        }
    }

    private static void deleteCloudFormationStack() {
        try {
            CloudFormationHelper.emptyS3Bucket(glueBucketName);
            CloudFormationHelper.destroyCloudFormationStack(STACK_NAME);
            logger.info("Resources deleted successfully!");
        } catch (CloudFormationException e) {
            throw new RuntimeException("Failed to delete CloudFormation stack: " +
 e.getMessage(), e);
        } catch (S3Exception e) {
            throw new RuntimeException("Failed to empty S3 bucket: " +
 e.getMessage(), e);
        }
    }
}
```

AWS Entity Resolution 数据匹配服务 SDK 方法的包装器类。

```
public class EntityResActions {

    private static final String PREFIX = "eroutput/";
    private static final Logger logger =
 LoggerFactory.getLogger(EntityResActions.class);
```

```
   private static EntityResolutionAsyncClient entityResolutionAsyncClient;

   private static S3AsyncClient s3AsyncClient;

   public static EntityResolutionAsyncClient getResolutionAsyncClient() {
       if (entityResolutionAsyncClient == null) {
           /*
           The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
version 2,
           and it is designed to provide a high-performance, asynchronous HTTP
client for interacting with AWS services.
            It uses the Netty framework to handle the underlying network
communication and the Java NIO API to
            provide a non-blocking, event-driven approach to HTTP requests and
responses.
            */

           SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
               .maxConcurrency(50)  // Adjust as needed.
               .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
               .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
               .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
               .build();

           ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
               .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
timeout.
               .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
individual call attempt timeout.
               .retryStrategy(RetryMode.STANDARD)
               .build();

           entityResolutionAsyncClient = EntityResolutionAsyncClient.builder()
               .httpClient(httpClient)
               .overrideConfiguration(overrideConfig)
               .build();
       }
       return entityResolutionAsyncClient;
   }

   public static S3AsyncClient getS3AsyncClient() {
```

```
        if (s3AsyncClient == null) {
            /*
            The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
version 2,
            and it is designed to provide a high-performance, asynchronous HTTP
client for interacting with AWS services.
             It uses the Netty framework to handle the underlying network
communication and the Java NIO API to
             provide a non-blocking, event-driven approach to HTTP requests and
responses.
             */

            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(50)  // Adjust as needed.
                .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
                .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
                .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
timeout.
                .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
individual call attempt timeout.
                .retryStrategy(RetryMode.STANDARD)
                .build();

            s3AsyncClient = S3AsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return s3AsyncClient;
    }

    /**
     * Deletes the schema mapping asynchronously.
     *
     * @param schemaName the name of the schema to delete
     * @return a {@link CompletableFuture} that completes when the schema mapping is
deleted successfully,
     * or throws a {@link RuntimeException} if the deletion fails
```

```
     */
    public CompletableFuture<DeleteSchemaMappingResponse>
deleteSchemaMappingAsync(String schemaName) {
        DeleteSchemaMappingRequest request = DeleteSchemaMappingRequest.builder()
            .schemaName(schemaName)
            .build();

        return getResolutionAsyncClient().deleteSchemaMapping(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    // Successfully deleted the schema mapping, log the success
message.
                    logger.info("Schema mapping '{}' deleted successfully.",
schemaName);
                } else {
                    // Ensure exception is not null before accessing its cause.
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while deleting the schema mapping.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The schema mapping was not
found to delete: " + schemaName, cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to delete schema mapping:
" + schemaName, exception);
                }
            });
    }

    /**
     * Lists the schema mappings associated with the current AWS account. This
method uses an asynchronous paginator to
     * retrieve the schema mappings, and prints the name of each schema mapping to
the console.
     */
    public void ListSchemaMappings() {
        ListSchemaMappingsRequest mappingsRequest =
ListSchemaMappingsRequest.builder()
            .build();
```

```
        ListSchemaMappingsPublisher paginator =
getResolutionAsyncClient().listSchemaMappingsPaginator(mappingsRequest);

        // Iterate through the pages of results
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            response.schemaList().forEach(schemaMapping ->
                logger.info("Schema Mapping Name: " + schemaMapping.schemaName())
            );
        });

        // Wait for the asynchronous operation to complete
        future.join();
    }

    /**
     * Asynchronously deletes a workflow with the specified name.
     *
     * @param workflowName the name of the workflow to be deleted
     * @return a {@link CompletableFuture} that completes when the workflow has been
deleted
     * @throws RuntimeException if the deletion of the workflow fails
     */
    public CompletableFuture<DeleteMatchingWorkflowResponse>
deleteMatchingWorkflowAsync(String workflowName) {
        DeleteMatchingWorkflowRequest request =
DeleteMatchingWorkflowRequest.builder()
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().deleteMatchingWorkflow(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("{} was deleted", workflowName );
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while deleting the workflow.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The workflow to delete was
not found.", cause);
```

```
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to delete workflow: " +
exception.getMessage(), exception);
                }
            });
    }

    /**
     * Creates a schema mapping asynchronously.
     *
     * @param schemaName the name of the schema to create
     * @return a {@link CompletableFuture} that represents the asynchronous creation
of the schema mapping
     */
    public CompletableFuture<CreateSchemaMappingResponse>
createSchemaMappingAsync(String schemaName) {
        List<SchemaInputAttribute> schemaAttributes = null;
        if (schemaName.startsWith("json")) {
            schemaAttributes = List.of(

SchemaInputAttribute.builder().matchKey("id").fieldName("id").type(SchemaAttributeType.UNIQ

SchemaInputAttribute.builder().matchKey("name").fieldName("name").type(SchemaAttributeType.

SchemaInputAttribute.builder().matchKey("email").fieldName("email").type(SchemaAttributeTyp
            );
        } else {
            schemaAttributes = List.of(

SchemaInputAttribute.builder().matchKey("id").fieldName("id").type(SchemaAttributeType.UNIQ

SchemaInputAttribute.builder().matchKey("name").fieldName("name").type(SchemaAttributeType.

SchemaInputAttribute.builder().matchKey("email").fieldName("email").type(SchemaAttributeTyp

SchemaInputAttribute.builder().fieldName("phone").type(SchemaAttributeType.PROVIDER_ID).sub
            );
        }

        CreateSchemaMappingRequest request = CreateSchemaMappingRequest.builder()
            .schemaName(schemaName)
            .mappedInputFields(schemaAttributes)
```

```
            .build();

        return getResolutionAsyncClient().createSchemaMapping(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("[{}] schema mapping Created Successfully!",
schemaName);
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while creating the schema mapping.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflicting schema mapping
already exists. Resolve conflicts before proceeding.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to create schema mapping:
" + exception.getMessage(), exception);
                }
            });
    }

    /**
     * Retrieves the schema mapping asynchronously.
     *
     * @param schemaName the name of the schema to retrieve the mapping for
     * @return a {@link CompletableFuture} that completes with the {@link
GetSchemaMappingResponse} when the operation
     * is complete
     * @throws RuntimeException if the schema mapping retrieval fails
     */
    public CompletableFuture<GetSchemaMappingResponse> getSchemaMappingAsync(String
schemaName) {
        GetSchemaMappingRequest mappingRequest = GetSchemaMappingRequest.builder()
            .schemaName(schemaName)
            .build();

        return getResolutionAsyncClient().getSchemaMapping(mappingRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
```

```
                    response.mappedInputFields().forEach(attribute ->
                        logger.info("Attribute Name: " + attribute.fieldName() +
                            ", Attribute Type: " + attribute.type().toString()));
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while getting schema mapping.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested schema mapping
was not found.", cause);
                    }

                    // Wrap other exceptions in a CompletionException with the
message.
                    throw new CompletionException("Failed to get schema mapping: " +
exception.getMessage(), exception);
                }
            });
    }

    /**
     * Asynchronously retrieves a matching job based on the provided job ID and
workflow name.
     *
     * @param jobId        the ID of the job to retrieve
     * @param workflowName the name of the workflow associated with the job
     * @return a {@link CompletableFuture} that completes when the job information
is available or an exception occurs
     */
    public CompletableFuture<GetMatchingJobResponse> getMatchingJobAsync(String
jobId, String workflowName) {
        GetMatchingJobRequest request = GetMatchingJobRequest.builder()
            .jobId(jobId)
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().getMatchingJob(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    // Successfully fetched the matching job details, log the job
status.
```

```
                    logger.info("Job status: " + response.status());
                    logger.info("Job details: " + response.toString());
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while fetching the matching job.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested job could not
be found.", cause);
                    }

                    // Wrap other exceptions in a CompletionException with the
message.
                    throw new CompletionException("Error fetching matching job: " +
exception.getMessage(), exception);
                }
            });
    }


    /**
     * Starts a matching job asynchronously for the specified workflow name.
     *
     * @param workflowName the name of the workflow for which to start the matching
job
     * @return a {@link CompletableFuture} that completes with the job ID of the
started matching job, or an empty
     * string if the operation fails
     */
    public CompletableFuture<String> startMatchingJobAsync(String workflowName) {
        StartMatchingJobRequest jobRequest = StartMatchingJobRequest.builder()
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().startMatchingJob(jobRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    String jobId = response.jobId();
                    logger.info("Job ID: " + jobId);
                } else {
                    if (exception == null) {
```

```
                        throw new CompletionException("An unknown error occurred
while starting the job.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("The job is already running.
Resolve conflicts before starting a new job.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to start the job: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response != null ? response.jobId() : "");
    }

    /**
     * Checks the status of a workflow asynchronously.
     *
     * @param jobId        the ID of the job to check
     * @param workflowName the name of the workflow to check
     * @return a CompletableFuture that resolves to a boolean value indicating
whether the workflow has completed
     * successfully
     */
    public CompletableFuture<GetMatchingJobResponse>
checkWorkflowStatusCompleteAsync(String jobId, String workflowName) {
        GetMatchingJobRequest request = GetMatchingJobRequest.builder()
            .jobId(jobId)
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().getMatchingJob(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    // Process the response and log the job status.
                    logger.info("Job status: " + response.status());
                } else {
                    // Ensure exception is not null before accessing its cause.
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while checking job status.", null);
```

```
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested resource was
not found while checking the job status.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to check job status: " +
exception.getMessage(), exception);
                }
            });
    }

    /**
     * Creates an asynchronous CompletableFuture to manage the creation of a
matching workflow.
     *
     * @param roleARN                the AWS IAM role ARN to be used for the
workflow execution
     * @param workflowName           the name of the workflow to be created
     * @param outputBucket           the S3 bucket path where the workflow output
will be stored
     * @param jsonGlueTableArn       the ARN of the Glue Data Catalog table to be
used as the input source
     * @param jsonErSchemaMappingName the name of the schema to be used for the
input source
     * @return a CompletableFuture that, when completed, will return the ARN of the
created workflow
     */
    public CompletableFuture<String> createMatchingWorkflowAsync(
        String roleARN
        , String workflowName
        , String outputBucket
        , String jsonGlueTableArn
        , String jsonErSchemaMappingName
        , String csvGlueTableArn
        , String csvErSchemaMappingName) {

        InputSource jsonInputSource = InputSource.builder()
            .inputSourceARN(jsonGlueTableArn)
            .schemaName(jsonErSchemaMappingName)
            .applyNormalization(false)
```

```
            .build();

        InputSource csvInputSource = InputSource.builder()
            .inputSourceARN(csvGlueTableArn)
            .schemaName(csvErSchemaMappingName)
            .applyNormalization(false)
            .build();

        OutputAttribute idOutputAttribute = OutputAttribute.builder()
            .name("id")
            .build();

        OutputAttribute nameOutputAttribute = OutputAttribute.builder()
            .name("name")
            .build();

        OutputAttribute emailOutputAttribute = OutputAttribute.builder()
            .name("email")
            .build();

        OutputAttribute phoneOutputAttribute = OutputAttribute.builder()
            .name("phone")
            .build();

        OutputSource outputSource = OutputSource.builder()
            .outputS3Path("s3://" + outputBucket + "/eroutput")
            .output(idOutputAttribute, nameOutputAttribute, emailOutputAttribute,
phoneOutputAttribute)
            .applyNormalization(false)
            .build();

        ResolutionTechniques resolutionType = ResolutionTechniques.builder()
            .resolutionType(ResolutionType.ML_MATCHING)
            .build();

        CreateMatchingWorkflowRequest workflowRequest =
CreateMatchingWorkflowRequest.builder()
            .roleArn(roleARN)
            .description("Created by using the AWS SDK for Java")
            .workflowName(workflowName)
            .inputSourceConfig(List.of(jsonInputSource, csvInputSource))
            .outputSourceConfig(List.of(outputSource))
            .resolutionTechniques(resolutionType)
            .build();
```

```
        return getResolutionAsyncClient().createMatchingWorkflow(workflowRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Workflow created successfully.");
                } else {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ValidationException) {
                        throw new CompletionException("Invalid request: Please check
input parameters.", cause);
                    }

                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflicting workflow
already exists. Resolve conflicts before proceeding.", cause);
                    }
                    throw new CompletionException("Failed to create workflow: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(CreateMatchingWorkflowResponse::workflowArn);
    }

    /**
     * Tags the specified schema mapping ARN.
     *
     * @param schemaMappingARN the ARN of the schema mapping to tag
     */
    public CompletableFuture<TagResourceResponse> tagEntityResource(String
schemaMappingARN) {
        Map<String, String> tags = new HashMap<>();
        tags.put("tag1", "tag1Value");
        tags.put("tag2", "tag2Value");

        TagResourceRequest request = TagResourceRequest.builder()
            .resourceArn(schemaMappingARN)
            .tags(tags)
            .build();

        return getResolutionAsyncClient().tagResource(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    // Successfully tagged the resource, log the success message.
                    logger.info("Successfully tagged the resource.");
```

```
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while tagging the resource.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The resource to tag was not
found.", cause);
                    }
                    throw new CompletionException("Failed to tag the resource: " +
exception.getMessage(), exception);
                }
            });
   }

   public CompletableFuture<JobMetrics> getJobInfo(String workflowName, String
jobId) {
       return getResolutionAsyncClient().getMatchingJob(b -> b
               .workflowName(workflowName)
               .jobId(jobId))
           .whenComplete((response, exception) -> {
               if (response != null) {
                   logger.info("Job metrics fetched successfully for jobId: " +
jobId);
               } else {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("Invalid request: Job id was
not found.", cause);
                   }
                   throw new CompletionException("Failed to fetch job info: " +
exception.getMessage(), exception);
               }
           })
           .thenApply(response -> response.metrics()); // Extract job metrics
   }

   /**
    * Uploads data to an Amazon S3 bucket asynchronously.
    *
    * @param bucketName the name of the S3 bucket to upload the data to
    * @param jsonData   the JSON data to be uploaded
```

```
     * @param csvData    the CSV data to be uploaded
     * @return a {@link CompletableFuture} representing both asynchronous operation
of uploading the data
     * @throws RuntimeException if an error occurs during the file upload
     */

   public void uploadInputData(String bucketName, String jsonData, String csvData)
{
       // Upload JSON data.
       String jsonKey = "jsonData/data.json";
       PutObjectRequest jsonUploadRequest = PutObjectRequest.builder()
           .bucket(bucketName)
           .key(jsonKey)
           .contentType("application/json")
           .build();

       CompletableFuture<PutObjectResponse> jsonUploadResponse =
getS3AsyncClient().putObject(jsonUploadRequest,
AsyncRequestBody.fromString(jsonData));

       // Upload CSV data.
       String csvKey = "csvData/data.csv";
       PutObjectRequest csvUploadRequest = PutObjectRequest.builder()
           .bucket(bucketName)
           .key(csvKey)
           .contentType("text/csv")
           .build();
       CompletableFuture<PutObjectResponse> csvUploadResponse =
getS3AsyncClient().putObject(csvUploadRequest,
AsyncRequestBody.fromString(csvData));

       CompletableFuture.allOf(jsonUploadResponse, csvUploadResponse)
           .whenComplete((result, ex) -> {
               if (ex != null) {
                   // Wrap an AWS exception.
                   throw new CompletionException("Failed to upload files", ex);
               }
           }).join();

   }

   /**
    * Finds the latest file in the S3 bucket that starts with "run-" in any depth
of subfolders
```

```
     */
    private CompletableFuture<String> findLatestMatchingFile(String bucketName) {
        ListObjectsV2Request request = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .prefix(PREFIX) // Searches within the given folder
            .build();

        return getS3AsyncClient().listObjectsV2(request)
            .thenApply(response -> response.contents().stream()
                .map(S3Object::key)
                .filter(key -> key.matches(".*?/run-[0-9a-zA-Z\\-]+")) // Matches
files like run-XXXXX in any subfolder
                .max(String::compareTo) // Gets the latest file
                .orElse(null))
            .whenComplete((result, exception) -> {
                if (exception == null) {
                    if (result != null) {
                        logger.info("Latest matching file found: " + result);
                    } else {
                        logger.info("No matching files found.");
                    }
                } else {
                    throw new CompletionException("Failed to find latest matching
file: " + exception.getMessage(), exception);
                }
            });
    }

    /**
     * Prints the data located in the file in the S3 bucket that starts with "run-"
in any depth of subfolders
     */
    public void printData(String bucketName) {
        try {
            // Find the latest file with "run-" prefix in any depth of subfolders.
            String s3Key = findLatestMatchingFile(bucketName).join();
            if (s3Key == null) {
                logger.error("No matching files found in S3.");
                return;
            }

            logger.info("Downloading file: " + s3Key);

            // Read CSV file as String.
```

```
            String csvContent = readCSVFromS3Async(bucketName, s3Key).join();
            if (csvContent.isEmpty()) {
                logger.error("File is empty.");
                return;
            }

            // Process CSV content.
            List<String[]> records = parseCSV(csvContent);
            printTable(records);

        } catch (RuntimeException | IOException | CsvException e) {
            logger.error("Error processing CSV file from S3: " + e.getMessage());
            e.printStackTrace();
        }
    }

    /**
     * Reads a CSV file from S3 and returns it as a String.
     */
    private static CompletableFuture<String> readCSVFromS3Async(String bucketName,
String s3Key) {
        GetObjectRequest getObjectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(s3Key)
            .build();

        // Initiating the asynchronous request to get the file as bytes
        return getS3AsyncClient().getObject(getObjectRequest,
AsyncResponseTransformer.toBytes())
            .thenApply(responseBytes -> responseBytes.asUtf8String()) // Convert
bytes to UTF-8 string
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    throw new CompletionException("Failed to read CSV from S3: " +
exception.getMessage(), exception);
                } else {
                    logger.info("Successfully fetched CSV file content from S3.");
                }
            });
    }

    /**
     * Parses CSV content from a String into a list of records.
     */
```

```
    private static List<String[]> parseCSV(String csvContent) throws IOException,
CsvException {
        try (CSVReader csvReader = new CSVReader(new StringReader(csvContent))) {
            return csvReader.readAll();
        }
    }

    /**
     * Prints the given CSV data in a formatted table
     */
    private static void printTable(List<String[]> records) {
        if (records.isEmpty()) {
            System.out.println("No records found.");
            return;
        }

        String[] headers = records.get(0);
        List<String[]> rows = records.subList(1, records.size());

        // Determine column widths dynamically based on longest content
        int[] columnWidths = new int[headers.length];
        for (int i = 0; i < headers.length; i++) {
            final int columnIndex = i;
            int maxWidth = Math.max(headers[i].length(), rows.stream()
                .map(row -> row.length > columnIndex ? row[columnIndex].length() :
0)
                .max(Integer::compareTo)
                .orElse(0));
            columnWidths[i] = Math.min(maxWidth, 25); // Limit max width for better
readability
        }

        // Enable ANSI Console for colored output
        AnsiConsole.systemInstall();

        // Print table header
        System.out.println(ansi().fgYellow().a("=== CSV Data from S3 ===").reset());
        printRow(headers, columnWidths, true);

        // Print rows
        rows.forEach(row -> printRow(row, columnWidths, false));

        // Restore console to normal
        AnsiConsole.systemUninstall();
```

```
    }

    private static void printRow(String[] row, int[] columnWidths, boolean isHeader)
 {
        String border = IntStream.range(0, columnWidths.length)
            .mapToObj(i -> "-".repeat(columnWidths[i] + 2))
            .collect(Collectors.joining("+", "+", "+"));

        if (isHeader) {
            System.out.println(border);
        }

        System.out.print("|");
        for (int i = 0; i < columnWidths.length; i++) {
            String cell = (i < row.length && row[i] != null) ? row[i] : "";
            System.out.printf(" %-" + columnWidths[i] + "s |", isHeader ?
 ansi().fgBrightBlue().a(cell).reset() : cell);
        }
        System.out.println();

        if (isHeader) {
            System.out.println(border);
        }
    }
}
```

操作

`CreateMatchingWorkflow`

以下代码示例演示了如何使用 `CreateMatchingWorkflow`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Creates an asynchronous CompletableFuture to manage the creation of a
matching workflow.
    *
    * @param roleARN                the AWS IAM role ARN to be used for the
workflow execution
    * @param workflowName           the name of the workflow to be created
    * @param outputBucket           the S3 bucket path where the workflow output
will be stored
    * @param jsonGlueTableArn       the ARN of the Glue Data Catalog table to be
used as the input source
    * @param jsonErSchemaMappingName the name of the schema to be used for the
input source
    * @return a CompletableFuture that, when completed, will return the ARN of the
created workflow
    */
   public CompletableFuture<String> createMatchingWorkflowAsync(
       String roleARN
       , String workflowName
       , String outputBucket
       , String jsonGlueTableArn
       , String jsonErSchemaMappingName
       , String csvGlueTableArn
       , String csvErSchemaMappingName) {

       InputSource jsonInputSource = InputSource.builder()
           .inputSourceARN(jsonGlueTableArn)
           .schemaName(jsonErSchemaMappingName)
           .applyNormalization(false)
           .build();

       InputSource csvInputSource = InputSource.builder()
           .inputSourceARN(csvGlueTableArn)
           .schemaName(csvErSchemaMappingName)
           .applyNormalization(false)
           .build();

       OutputAttribute idOutputAttribute = OutputAttribute.builder()
           .name("id")
           .build();

       OutputAttribute nameOutputAttribute = OutputAttribute.builder()
           .name("name")
```

```
            .build();

        OutputAttribute emailOutputAttribute = OutputAttribute.builder()
            .name("email")
            .build();

        OutputAttribute phoneOutputAttribute = OutputAttribute.builder()
            .name("phone")
            .build();

        OutputSource outputSource = OutputSource.builder()
            .outputS3Path("s3://" + outputBucket + "/eroutput")
            .output(idOutputAttribute, nameOutputAttribute, emailOutputAttribute,
phoneOutputAttribute)
            .applyNormalization(false)
            .build();

        ResolutionTechniques resolutionType = ResolutionTechniques.builder()
            .resolutionType(ResolutionType.ML_MATCHING)
            .build();

        CreateMatchingWorkflowRequest workflowRequest =
CreateMatchingWorkflowRequest.builder()
            .roleArn(roleARN)
            .description("Created by using the AWS SDK for Java")
            .workflowName(workflowName)
            .inputSourceConfig(List.of(jsonInputSource, csvInputSource))
            .outputSourceConfig(List.of(outputSource))
            .resolutionTechniques(resolutionType)
            .build();

        return getResolutionAsyncClient().createMatchingWorkflow(workflowRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Workflow created successfully.");
                } else {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ValidationException) {
                        throw new CompletionException("Invalid request: Please check
input parameters.", cause);
                    }

                    if (cause instanceof ConflictException) {
```

```
                        throw new CompletionException("A conflicting workflow
already exists. Resolve conflicts before proceeding.", cause);
                    }
                    throw new CompletionException("Failed to create workflow: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(CreateMatchingWorkflowResponse::workflowArn);
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateMatchingWorkflow中的。

`CreateSchemaMapping`

以下代码示例演示了如何使用 `CreateSchemaMapping`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Creates a schema mapping asynchronously.
     *
     * @param schemaName the name of the schema to create
     * @return a {@link CompletableFuture} that represents the asynchronous creation
of the schema mapping
     */
    public CompletableFuture<CreateSchemaMappingResponse>
createSchemaMappingAsync(String schemaName) {
        List<SchemaInputAttribute> schemaAttributes = null;
        if (schemaName.startsWith("json")) {
            schemaAttributes = List.of(

SchemaInputAttribute.builder().matchKey("id").fieldName("id").type(SchemaAttributeType.UNIQ

SchemaInputAttribute.builder().matchKey("name").fieldName("name").type(SchemaAttributeType.
```

```
SchemaInputAttribute.builder().matchKey("email").fieldName("email").type(SchemaAttributeTyp
            );
        } else {
            schemaAttributes = List.of(

SchemaInputAttribute.builder().matchKey("id").fieldName("id").type(SchemaAttributeType.UNIQ

SchemaInputAttribute.builder().matchKey("name").fieldName("name").type(SchemaAttributeType.

SchemaInputAttribute.builder().matchKey("email").fieldName("email").type(SchemaAttributeTyp

SchemaInputAttribute.builder().fieldName("phone").type(SchemaAttributeType.PROVIDER_ID).sub
            );
        }

        CreateSchemaMappingRequest request = CreateSchemaMappingRequest.builder()
            .schemaName(schemaName)
            .mappedInputFields(schemaAttributes)
            .build();

        return getResolutionAsyncClient().createSchemaMapping(request)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("[{}] schema mapping Created Successfully!",
schemaName);
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while creating the schema mapping.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflicting schema mapping
already exists. Resolve conflicts before proceeding.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to create schema mapping:
" + exception.getMessage(), exception);
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateSchemaMapping中的。

`DeleteMatchingWorkflow`

以下代码示例演示了如何使用 `DeleteMatchingWorkflow`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Asynchronously deletes a workflow with the specified name.
    *
    * @param workflowName the name of the workflow to be deleted
    * @return a {@link CompletableFuture} that completes when the workflow has been
deleted
    * @throws RuntimeException if the deletion of the workflow fails
    */
   public CompletableFuture<DeleteMatchingWorkflowResponse>
deleteMatchingWorkflowAsync(String workflowName) {
       DeleteMatchingWorkflowRequest request =
DeleteMatchingWorkflowRequest.builder()
           .workflowName(workflowName)
           .build();

       return getResolutionAsyncClient().deleteMatchingWorkflow(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   logger.info("{} was deleted", workflowName );
               } else {
                   if (exception == null) {
                       throw new CompletionException("An unknown error occurred
while deleting the workflow.", null);
                   }

                   Throwable cause = exception.getCause();
```

```
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The workflow to delete was
not found.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to delete workflow: " +
exception.getMessage(), exception);
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteMatchingWorkflow中的。

`DeleteSchemaMapping`

以下代码示例演示了如何使用 `DeleteSchemaMapping`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Deletes the schema mapping asynchronously.
     *
     * @param schemaName the name of the schema to delete
     * @return a {@link CompletableFuture} that completes when the schema mapping is
deleted successfully,
     * or throws a {@link RuntimeException} if the deletion fails
     */
    public CompletableFuture<DeleteSchemaMappingResponse>
deleteSchemaMappingAsync(String schemaName) {
        DeleteSchemaMappingRequest request = DeleteSchemaMappingRequest.builder()
            .schemaName(schemaName)
            .build();

        return getResolutionAsyncClient().deleteSchemaMapping(request)
```

```
            .whenComplete((response, exception) -> {
                if (response != null) {
                    // Successfully deleted the schema mapping, log the success
message.
                    logger.info("Schema mapping '{}' deleted successfully.",
schemaName);
                } else {
                    // Ensure exception is not null before accessing its cause.
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while deleting the schema mapping.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The schema mapping was not
found to delete: " + schemaName, cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to delete schema mapping:
" + schemaName, exception);
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[DeleteSchemaMapping](https://docs.aws.amazon.com/goto/SdkForJavaV2/entityresolution-2018-05-10/DeleteSchemaMapping)中的。

`GetMatchingJob`

以下代码示例演示了如何使用 `GetMatchingJob`。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 [AWS 代码示例存储库](https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/javav2/example_code/entityresolution#readme)中查找完整示例，了解如何进行设置和运行。

```
    /**
```

```
    * Asynchronously retrieves a matching job based on the provided job ID and
workflow name.
    *
    * @param jobId        the ID of the job to retrieve
    * @param workflowName the name of the workflow associated with the job
    * @return a {@link CompletableFuture} that completes when the job information
is available or an exception occurs
    */
   public CompletableFuture<GetMatchingJobResponse> getMatchingJobAsync(String
jobId, String workflowName) {
       GetMatchingJobRequest request = GetMatchingJobRequest.builder()
           .jobId(jobId)
           .workflowName(workflowName)
           .build();

       return getResolutionAsyncClient().getMatchingJob(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   // Successfully fetched the matching job details, log the job
status.
                   logger.info("Job status: " + response.status());
                   logger.info("Job details: " + response.toString());
               } else {
                   if (exception == null) {
                       throw new CompletionException("An unknown error occurred
while fetching the matching job.", null);
                   }

                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The requested job could not
be found.", cause);
                   }

                   // Wrap other exceptions in a CompletionException with the
message.
                   throw new CompletionException("Error fetching matching job: " +
exception.getMessage(), exception);
               }
           });
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetMatchingJob中的。

GetSchemaMapping

以下代码示例演示了如何使用 GetSchemaMapping。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Retrieves the schema mapping asynchronously.
    *
    * @param schemaName the name of the schema to retrieve the mapping for
    * @return a {@link CompletableFuture} that completes with the {@link
GetSchemaMappingResponse} when the operation
    * is complete
    * @throws RuntimeException if the schema mapping retrieval fails
    */
   public CompletableFuture<GetSchemaMappingResponse> getSchemaMappingAsync(String
schemaName) {
       GetSchemaMappingRequest mappingRequest = GetSchemaMappingRequest.builder()
           .schemaName(schemaName)
           .build();

       return getResolutionAsyncClient().getSchemaMapping(mappingRequest)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   response.mappedInputFields().forEach(attribute ->
                       logger.info("Attribute Name: " + attribute.fieldName() +
                           ", Attribute Type: " + attribute.type().toString()));
               } else {
                   if (exception == null) {
                       throw new CompletionException("An unknown error occurred
while getting schema mapping.", null);
                   }

                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The requested schema mapping
was not found.", cause);
```

```
                    }

                    // Wrap other exceptions in a CompletionException with the
message.
                    throw new CompletionException("Failed to get schema mapping: " +
exception.getMessage(), exception);
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetSchemaMapping中的。

`ListSchemaMappings`

以下代码示例演示了如何使用 `ListSchemaMappings`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Lists the schema mappings associated with the current AWS account. This
method uses an asynchronous paginator to
     * retrieve the schema mappings, and prints the name of each schema mapping to
the console.
     */
    public void ListSchemaMappings() {
        ListSchemaMappingsRequest mappingsRequest =
ListSchemaMappingsRequest.builder()
            .build();

        ListSchemaMappingsPublisher paginator =
getResolutionAsyncClient().listSchemaMappingsPaginator(mappingsRequest);

        // Iterate through the pages of results
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            response.schemaList().forEach(schemaMapping ->
```

```
                logger.info("Schema Mapping Name: " + schemaMapping.schemaName())
            );
        });

        // Wait for the asynchronous operation to complete
        future.join();
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ListSchemaMappings](#)中的。

`StartMatchingJob`

以下代码示例演示了如何使用 `StartMatchingJob`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Starts a matching job asynchronously for the specified workflow name.
     *
     * @param workflowName the name of the workflow for which to start the matching
job
     * @return a {@link CompletableFuture} that completes with the job ID of the
started matching job, or an empty
     * string if the operation fails
     */
    public CompletableFuture<String> startMatchingJobAsync(String workflowName) {
        StartMatchingJobRequest jobRequest = StartMatchingJobRequest.builder()
            .workflowName(workflowName)
            .build();

        return getResolutionAsyncClient().startMatchingJob(jobRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    String jobId = response.jobId();
```

```
                    logger.info("Job ID: " + jobId);
                } else {
                    if (exception == null) {
                        throw new CompletionException("An unknown error occurred
while starting the job.", null);
                    }

                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("The job is already running.
Resolve conflicts before starting a new job.", cause);
                    }

                    // Wrap other AWS exceptions in a CompletionException.
                    throw new CompletionException("Failed to start the job: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response != null ? response.jobId() : "");
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartMatchingJob中的。

`TagResource`

以下代码示例演示了如何使用 `TagResource`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Tags the specified schema mapping ARN.
     *
     * @param schemaMappingARN the ARN of the schema mapping to tag
     */
```

```
   public CompletableFuture<TagResourceResponse> tagEntityResource(String
schemaMappingARN) {
       Map<String, String> tags = new HashMap<>();
       tags.put("tag1", "tag1Value");
       tags.put("tag2", "tag2Value");

       TagResourceRequest request = TagResourceRequest.builder()
           .resourceArn(schemaMappingARN)
           .tags(tags)
           .build();

       return getResolutionAsyncClient().tagResource(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   // Successfully tagged the resource, log the success message.
                   logger.info("Successfully tagged the resource.");
               } else {
                   if (exception == null) {
                       throw new CompletionException("An unknown error occurred
while tagging the resource.", null);
                   }

                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The resource to tag was not
found.", cause);
                   }
                   throw new CompletionException("Failed to tag the resource: " +
exception.getMessage(), exception);
               }
           });
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[TagResource](#)中的。

OpenSearch 使用适用于 Java 的 SDK 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

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 基本功能
- 操作

开始使用

你好 OpenSearch 服务

以下代码示例显示了如何开始使用 S OpenSearch ervice。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.services.opensearch.OpenSearchAsyncClient;
import software.amazon.awssdk.services.opensearch.model.ListVersionsRequest;
import java.util.List;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloOpenSearch {
    public static void main(String[] args) {
```

```
        try {
            CompletableFuture<Void> future = listVersionsAsync();
            future.join();
            System.out.println("Versions listed successfully.");
        } catch (RuntimeException e) {
            System.err.println("Error occurred while listing versions: " +
 e.getMessage());
        }
    }

    private static OpenSearchAsyncClient getAsyncClient() {
        return OpenSearchAsyncClient.builder().build();
    }

    public static CompletableFuture<Void> listVersionsAsync() {
        ListVersionsRequest request = ListVersionsRequest.builder()
            .maxResults(10)
            .build();

        return getAsyncClient().listVersions(request).thenAccept(response -> {
            List<String> versionList = response.versions();
            for (String version : versionList) {
                System.out.println("Version info: " + version);
            }
        }).exceptionally(ex -> {
            // Handle the exception, or propagate it as a RuntimeException
            throw new RuntimeException("Failed to list versions", ex);
        });
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListVersions中的。

基本功能

学习 OpenSearch 服务核心运营

以下代码示例展示了如何：

- 创建 OpenSearch 服务域。
- 提供有关特定 OpenSearch 服务域的详细信息。

- 列出该账户拥有的所有 OpenSearch 服务域。
- 等待，直到 OpenSearch 服务域的更改状态达到已完成状态。
- 修改现有 OpenSearch 服务域的配置。
- 向 OpenSearch 服务域添加标签。
- 列出与 OpenSearch 服务域关联的标签。
- 从 OpenSearch 服务域中移除标签。
- 删除 OpenSearch 服务域。

适用于 Java 的 SDK 2.x

> **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。

运行演示 OpenSearch 服务功能的交互式场景。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.opensearch.model.*;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;

public class OpenSearchScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    private static final Logger logger =
 LoggerFactory.getLogger(OpenSearchScenario.class);
    static Scanner scanner = new Scanner(System.in);

    static OpenSearchActions openSearchActions = new OpenSearchActions();

    public static void main(String[] args) throws Throwable {
        logger.info("""
            Welcome to the Amazon OpenSearch Service Basics Scenario.
```

```
            Use the Amazon OpenSearch Service API to create, configure, and manage
 OpenSearch Service domains.

            The operations exposed by the AWS OpenSearch Service client are focused
 on managing the OpenSearch Service domains
            and their configurations, not the data within the domains (such as
 indexing or querying documents).
            For document management, you typically interact directly with the
 OpenSearch REST API or use other libraries,
            such as the OpenSearch Java client (https://opensearch.org/docs/latest/
clients/java/).

            Let's get started...
        """);
        waitForInputToContinue(scanner);
        try {
            runScenario();
        } catch (RuntimeException e) {
            e.printStackTrace();
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }

    private static void runScenario() throws Throwable {
        String currentTimestamp = String.valueOf(System.currentTimeMillis());
        String domainName = "test-domain-" + currentTimestamp;

        logger.info(DASHES);
        logger.info("1. Create an Amazon OpenSearch domain");
```

```
        logger.info("""
            An Amazon OpenSearch domain is a managed instance of the OpenSearch
engine,
            which is an open-source search and analytics engine derived from
Elasticsearch.
            An OpenSearch domain is essentially a cluster of compute resources and
storage that hosts
            one or more OpenSearch indexes, enabling you to perform full-text
searches, data analysis, and
            visualizations.

            In this step, we'll initiate the creation of the domain. We'll check on
the progress in a later step.
        """);
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<String> future =
openSearchActions.createNewDomainAsync(domainName);
            String domainId = future.join();
            logger.info("Domain successfully created with ID: {}", domainId);
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause != null) {
                if (cause instanceof OpenSearchException openSearchEx) {
                    logger.error("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
                } else {
                    logger.error("An unexpected error occurred: " +
cause.getMessage(), cause);
                }
            } else {
                logger.error("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("2. Describe the Amazon OpenSearch domain");
        logger.info("In this step, we get back the Domain ARN which is used in an
upcoming step.");
        waitForInputToContinue(scanner);
```

```
        String arn = "";
        try {
            CompletableFuture<String> future =
openSearchActions.describeDomainAsync(domainName);
            arn = future.join();
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("3. List the domains in your account");
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<List<DomainInfo>> future =
openSearchActions.listAllDomainsAsync();
            List<DomainInfo> domainInfoList = future.join();
            for (DomainInfo domain : domainInfoList) {
                logger.info("Domain name is: " + domain.domainName());
            }
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
OpenSearchException)) {
                cause = cause.getCause();
            }
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
```

```
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("4. Wait until the domain's change status reaches a completed
state");
        logger.info("""
            In this step, we check on the change status of the domain that we
initiated in Step 1.
            Until we reach a COMPLETED state, we stay in a loop by sending a
DescribeDomainChangeProgressRequest.

            The time it takes for a change to an OpenSearch domain to reach a
completed state can range
            from a few minutes to several hours. In this case the change is creating
a new domain that we initiated in Step 1.
            The time varies depending on the complexity of the change and the
current load on
            the OpenSearch service. In general, simple changes, such as scaling the
number of data nodes or
            updating the OpenSearch version, may take 10-30 minutes.
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<Void> future =
openSearchActions.domainChangeProgressAsync(domainName);
            future.join();
            logger.info("Domain change progress completed successfully.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
ResourceNotFoundException)) {
                cause = cause.getCause();
            }
            if (cause instanceof ResourceNotFoundException
resourceNotFoundException) {
                logger.info("The specific AWS resource was not found: Error message:
{}, Error code {}", resourceNotFoundException.awsErrorDetails().errorMessage(),
resourceNotFoundException.awsErrorDetails().errorCode());

                if (cause instanceof OpenSearchException ex) {
```

```
                    logger.info("An OpenSearch error occurred: Error message: " +
ex.getMessage());
                } else {
                    logger.info("An unexpected error occurred: " + rt.getMessage());
                }
                throw cause;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("5. Modify the domain");
        logger.info("""
            You can change your OpenSearch domain's settings, like the number of
instances, without starting over from scratch.
            This makes it easy to adjust your domain as your needs change, allowing
you to scale up or
            down quickly without recreating everything.

            We modify the domain in this step by changing the number of instances.
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<UpdateDomainConfigResponse> future =
openSearchActions.updateSpecificDomainAsync(domainName);
            UpdateDomainConfigResponse updateResponse = future.join();
            logger.info("Domain update status: " +
updateResponse.domainConfig().changeProgressDetails().configChangeStatusAsString());
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);
```

```
        logger.info("6. Wait until the domain's change status reaches a completed
state");
        logger.info("""
            In this step, we poll the status until the domain's change status
reaches a completed state.
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<Void> future =
openSearchActions.domainChangeProgressAsync(domainName);
            future.join();
            logger.info("Domain change progress completed successfully.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof OpenSearchException ex) {
                logger.info("EC2 error occurred: Error message: " +ex.getMessage());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("7. Tag the Domain");
        logger.info("""
            Tags let you assign arbitrary information to an Amazon OpenSearch
Service domain so you can
            categorize and filter on that information. A tag is a key-value pair
that you define and
            associate with an OpenSearch Service domain. You can use these tags to
track costs by grouping
            expenses for similarly tagged resources.

            In this scenario, we create tags with keys "service" and "instances".
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<AddTagsResponse> future =
openSearchActions.addDomainTagsAsync(arn);
```

```
            future.join();
            logger.info("Domain tags added successfully.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
OpenSearchException)) {
                cause = cause.getCause();
            }
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
                if (cause != null) {
                    if (cause instanceof OpenSearchException) {
                        logger.error("OpenSearch error occurred: Error message: " +
cause.getMessage(), cause);
                    } else {
                        logger.error("An unexpected error occurred: " +
cause.getMessage(), cause);
                    }
                } else {
                    logger.error("An unexpected error occurred: " + rt.getMessage(),
rt);
                }
                throw cause;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("8. List Domain tags");
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<ListTagsResponse> future =
openSearchActions.listDomainTagsAsync(arn);
            ListTagsResponse listTagsResponse = future.join();
            listTagsResponse.tagList().forEach(tag -> logger.info("Tag Key: " +
tag.key() + ", Tag Value: " + tag.value()));
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
```

```
            while (cause.getCause() != null && !(cause instanceof
OpenSearchException)) {
                cause = cause.getCause();
            }
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;

        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("9. Delete the domain");
        logger.info("""
            In this step, we'll delete the Amazon OpenSearch domain that we created
in Step 1.
            Deleting a domain will remove all data and configuration for that
domain.
        """);

        waitForInputToContinue(scanner);

        try {
            CompletableFuture<DeleteDomainResponse> future =
openSearchActions.deleteSpecificDomainAsync(domainName);
            future.join();
            logger.info("Domain successfully deleted.");
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            while (cause.getCause() != null && !(cause instanceof
OpenSearchException)) {
                cause = cause.getCause();
            }
            if (cause instanceof OpenSearchException openSearchEx) {
                logger.info("OpenSearch error occurred: Error message:
{}, Error code {}", openSearchEx.awsErrorDetails().errorMessage(),
openSearchEx.awsErrorDetails().errorCode());
            } else {
```

```
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;

        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("Scenario complete!");
    }
}
```

OpenSearch 服务 SDK 方法的包装器类。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.opensearch.OpenSearchAsyncClient;
import software.amazon.awssdk.services.opensearch.model.AddTagsRequest;
import software.amazon.awssdk.services.opensearch.model.AddTagsResponse;
import software.amazon.awssdk.services.opensearch.model.ClusterConfig;
import software.amazon.awssdk.services.opensearch.model.CreateDomainRequest;
import software.amazon.awssdk.services.opensearch.model.DeleteDomainRequest;
import software.amazon.awssdk.services.opensearch.model.DeleteDomainResponse;
import
 software.amazon.awssdk.services.opensearch.model.DescribeDomainChangeProgressRequest;
import
 software.amazon.awssdk.services.opensearch.model.DescribeDomainChangeProgressResponse;
import software.amazon.awssdk.services.opensearch.model.DescribeDomainRequest;
import software.amazon.awssdk.services.opensearch.model.DomainInfo;
import software.amazon.awssdk.services.opensearch.model.DomainStatus;
import software.amazon.awssdk.services.opensearch.model.EBSOptions;
import software.amazon.awssdk.services.opensearch.model.ListDomainNamesRequest;
import software.amazon.awssdk.services.opensearch.model.ListTagsRequest;
import software.amazon.awssdk.services.opensearch.model.ListTagsResponse;
import software.amazon.awssdk.services.opensearch.model.NodeToNodeEncryptionOptions;
import software.amazon.awssdk.services.opensearch.model.Tag;
import software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigRequest;
```

```
import software.amazon.awssdk.services.opensearch.model.UpdateDomainConfigResponse;
import software.amazon.awssdk.services.opensearch.model.VolumeType;
import java.time.Duration;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CompletableFuture;

public class OpenSearchActions {
    private static final Logger logger =
 LoggerFactory.getLogger(OpenSearchActions.class);
    private static OpenSearchAsyncClient openSearchClientAsyncClient;
    private static OpenSearchAsyncClient getAsyncClient() {
        if (openSearchClientAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryPolicy(RetryPolicy.builder()
                    .numRetries(3)
                    .build())
                .build();

            openSearchClientAsyncClient = OpenSearchAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return openSearchClientAsyncClient;
    }

    /**
     * Creates a new OpenSearch domain asynchronously.
     * @param domainName the name of the new OpenSearch domain to create
     * @return a {@link CompletableFuture} containing the domain ID of the newly
 created domain
     */
```

```
    public CompletableFuture<String> createNewDomainAsync(String domainName) {
        ClusterConfig clusterConfig = ClusterConfig.builder()
            .dedicatedMasterEnabled(true)
            .dedicatedMasterCount(3)
            .dedicatedMasterType("t2.small.search")
            .instanceType("t2.small.search")
            .instanceCount(5)
            .build();

        EBSOptions ebsOptions = EBSOptions.builder()
            .ebsEnabled(true)
            .volumeSize(10)
            .volumeType(VolumeType.GP2)
            .build();

        NodeToNodeEncryptionOptions encryptionOptions =
NodeToNodeEncryptionOptions.builder()
            .enabled(true)
            .build();

        CreateDomainRequest domainRequest = CreateDomainRequest.builder()
            .domainName(domainName)
            .engineVersion("OpenSearch_1.0")
            .clusterConfig(clusterConfig)
            .ebsOptions(ebsOptions)
            .nodeToNodeEncryptionOptions(encryptionOptions)
            .build();
        logger.info("Sending domain creation request...");
        return getAsyncClient().createDomain(domainRequest)
                .handle( (createResponse, throwable) -> {
                    if (createResponse != null) {
                        logger.info("Domain status is {}",
createResponse.domainStatus().changeProgressDetails().configChangeStatusAsString());
                        logger.info("Domain Id is {}",
createResponse.domainStatus().domainId());
                        return createResponse.domainStatus().domainId();
                    }
                    throw new RuntimeException("Failed to create domain",
throwable);
                });
    }

    /**
     * Deletes a specific domain asynchronously.
```

```
    * @param domainName the name of the domain to be deleted
    * @return a {@link CompletableFuture} that completes when the domain has been
deleted
    * or throws a {@link RuntimeException} if the deletion fails
    */
   public CompletableFuture<DeleteDomainResponse> deleteSpecificDomainAsync(String
domainName) {
       DeleteDomainRequest domainRequest = DeleteDomainRequest.builder()
           .domainName(domainName)
           .build();

       // Delete domain asynchronously
       return getAsyncClient().deleteDomain(domainRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   throw new RuntimeException("Failed to delete the domain: " +
domainName, exception);
               }
           });
   }

   /**
    * Describes the specified domain asynchronously.
    *
    * @param domainName the name of the domain to describe
    * @return a {@link CompletableFuture} that completes with the ARN of the domain
    * @throws RuntimeException if the domain description fails
    */
   public CompletableFuture<String> describeDomainAsync(String domainName) {
       DescribeDomainRequest request = DescribeDomainRequest.builder()
           .domainName(domainName)
           .build();

       return getAsyncClient().describeDomain(request)
           .handle((response, exception) -> {  // Handle both response and
exception
               if (exception != null) {
                   throw new RuntimeException("Failed to describe domain",
exception);
               }
               DomainStatus domainStatus = response.domainStatus();
               String endpoint = domainStatus.endpoint();
               String arn = domainStatus.arn();
               String engineVersion = domainStatus.engineVersion();
```

```
                logger.info("Domain endpoint is: " + endpoint);
                logger.info("ARN: " + arn);
                System.out.println("Engine version: " + engineVersion);

                return arn;  // Return ARN when successful
            });
    }

    /**
     * Asynchronously lists all the domains in the current AWS account.
     * @return a {@link CompletableFuture} that, when completed, contains a list of
{@link DomainInfo} objects representing
     *         the domains in the account.
     * @throws RuntimeException if there was a failure while listing the domains.
     */
    public CompletableFuture<List<DomainInfo>> listAllDomainsAsync() {
        ListDomainNamesRequest namesRequest = ListDomainNamesRequest.builder()
            .engineType("OpenSearch")
            .build();

        return getAsyncClient().listDomainNames(namesRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to list all domains",
exception);
                }
                return response.domainNames();  // Return the list of domain names
on success
            });
    }

    /**
     * Updates the configuration of a specific domain asynchronously.
     * @param domainName the name of the domain to update
     * @return a {@link CompletableFuture} that represents the asynchronous
operation of updating the domain configuration
     */
    public CompletableFuture<UpdateDomainConfigResponse>
updateSpecificDomainAsync(String domainName) {
        ClusterConfig clusterConfig = ClusterConfig.builder()
            .instanceCount(3)
            .build();
```

```
        UpdateDomainConfigRequest updateDomainConfigRequest =
UpdateDomainConfigRequest.builder()
            .domainName(domainName)
            .clusterConfig(clusterConfig)
            .build();

        return getAsyncClient().updateDomainConfig(updateDomainConfigRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to update the domain
configuration", exception);
                }
                // Handle success if needed (e.g., logging or additional actions)
            });
    }

    /**
     * Asynchronously checks the progress of a domain change operation in Amazon
OpenSearch Service.
     * @param domainName the name of the OpenSearch domain to check the progress for
     * @return a {@link CompletableFuture} that completes when the domain change
operation is completed
     */
    public CompletableFuture<Void> domainChangeProgressAsync(String domainName) {
        DescribeDomainChangeProgressRequest request =
DescribeDomainChangeProgressRequest.builder()
            .domainName(domainName)
            .build();

        return CompletableFuture.runAsync(() -> {
            boolean isCompleted = false;
            long startTime = System.currentTimeMillis();

            while (!isCompleted) {
                try {
                    // Handle the async client call using `join` to block
synchronously for the result
                    DescribeDomainChangeProgressResponse response = getAsyncClient()
                        .describeDomainChangeProgress(request)
                        .handle((resp, ex) -> {
                            if (ex != null) {
                                throw new RuntimeException("Failed to check domain
progress", ex);
                            }
```

```
                            return resp;
                        }).join();

                    String state = response.changeProgressStatus().statusAsString();
 // Get the status as string

                    if ("COMPLETED".equals(state)) {
                        logger.info("\nOpenSearch domain status: Completed");
                        isCompleted = true;
                    } else {
                        for (int i = 0; i < 5; i++) {
                            long elapsedTimeInSeconds = (System.currentTimeMillis()
- startTime) / 1000;
                            String formattedTime = String.format("%02d:%02d",
elapsedTimeInSeconds / 60, elapsedTimeInSeconds % 60);
                            System.out.print("\rOpenSearch domain state: " + state +
" | Time Elapsed: " + formattedTime + " ");
                            System.out.flush();
                            Thread.sleep(1_000);
                        }
                    }
                } catch (InterruptedException e) {
                    Thread.currentThread().interrupt();
                    throw new RuntimeException("Thread was interrupted", e);
                }
            }
        });
    }

    /**
     * Asynchronously adds tags to an Amazon OpenSearch Service domain.
     * @param domainARN the Amazon Resource Name (ARN) of the Amazon OpenSearch
Service domain to add tags to
     * @return a {@link CompletableFuture} that completes when the tags have been
successfully added to the domain,
     * or throws a {@link RuntimeException} if the operation fails
     */
    public CompletableFuture<AddTagsResponse> addDomainTagsAsync(String domainARN) {
        Tag tag1 = Tag.builder()
            .key("service")
            .value("OpenSearch")
            .build();

        Tag tag2 = Tag.builder()
```

```
            .key("instances")
            .value("m3.2xlarge")
            .build();

        List<Tag> tagList = new ArrayList<>();
        tagList.add(tag1);
        tagList.add(tag2);

        AddTagsRequest addTagsRequest = AddTagsRequest.builder()
            .arn(domainARN)
            .tagList(tagList)
            .build();

        return getAsyncClient().addTags(addTagsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to add tags to the domain: "
+ domainARN, exception);
                } else {
                    logger.info("Added Tags");
                }
            });
    }


    /**
     * Asynchronously lists the tags associated with the specified Amazon Resource
Name (ARN).
     * @param arn the Amazon Resource Name (ARN) of the resource for which to list
the tags
     * @return a {@link CompletableFuture} that, when completed, will contain a list
of the tags associated with the
     * specified ARN
     * @throws RuntimeException if there is an error listing the tags
     */
    public CompletableFuture<ListTagsResponse> listDomainTagsAsync(String arn) {
        ListTagsRequest tagsRequest = ListTagsRequest.builder()
            .arn(arn)
            .build();

        return getAsyncClient().listTags(tagsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
```

```
                    throw new RuntimeException("Failed to list domain tags",
 exception);
                }

                List<Tag> tagList = response.tagList();
                for (Tag tag : tagList) {
                    logger.info("Tag key is " + tag.key());
                    logger.info("Tag value is " + tag.value());
                }
            });
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - AddTags
  - CreateDomain
  - DeleteDomain
  - DescribeDomain
  - DescribeDomainChangeProgress
  - ListDomainNames
  - ListTags
  - UpdateDomainConfig

操作

`AddTags`

以下代码示例演示了如何使用 `AddTags`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Asynchronously adds tags to an Amazon OpenSearch Service domain.
     * @param domainARN the Amazon Resource Name (ARN) of the Amazon OpenSearch
Service domain to add tags to
     * @return a {@link CompletableFuture} that completes when the tags have been
successfully added to the domain,
     * or throws a {@link RuntimeException} if the operation fails
     */
    public CompletableFuture<AddTagsResponse> addDomainTagsAsync(String domainARN) {
        Tag tag1 = Tag.builder()
            .key("service")
            .value("OpenSearch")
            .build();

        Tag tag2 = Tag.builder()
            .key("instances")
            .value("m3.2xlarge")
            .build();

        List<Tag> tagList = new ArrayList<>();
        tagList.add(tag1);
        tagList.add(tag2);

        AddTagsRequest addTagsRequest = AddTagsRequest.builder()
            .arn(domainARN)
            .tagList(tagList)
            .build();

        return getAsyncClient().addTags(addTagsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to add tags to the domain: "
+ domainARN, exception);
                } else {
                    logger.info("Added Tags");
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考AddTags中的。

ChangeProgress

以下代码示例演示了如何使用 ChangeProgress。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Asynchronously checks the progress of a domain change operation in Amazon
OpenSearch Service.
    * @param domainName the name of the OpenSearch domain to check the progress for
    * @return a {@link CompletableFuture} that completes when the domain change
operation is completed
    */
   public CompletableFuture<Void> domainChangeProgressAsync(String domainName) {
       DescribeDomainChangeProgressRequest request =
DescribeDomainChangeProgressRequest.builder()
           .domainName(domainName)
           .build();

       return CompletableFuture.runAsync(() -> {
           boolean isCompleted = false;
           long startTime = System.currentTimeMillis();

           while (!isCompleted) {
               try {
                   // Handle the async client call using `join` to block
synchronously for the result
                   DescribeDomainChangeProgressResponse response = getAsyncClient()
                       .describeDomainChangeProgress(request)
                       .handle((resp, ex) -> {
                           if (ex != null) {
                               throw new RuntimeException("Failed to check domain
progress", ex);
                           }
                           return resp;
                       }).join();
```

```
                    String state = response.changeProgressStatus().statusAsString();
 // Get the status as string

                    if ("COMPLETED".equals(state)) {
                        logger.info("\nOpenSearch domain status: Completed");
                        isCompleted = true;
                    } else {
                        for (int i = 0; i < 5; i++) {
                            long elapsedTimeInSeconds = (System.currentTimeMillis()
- startTime) / 1000;
                            String formattedTime = String.format("%02d:%02d",
elapsedTimeInSeconds / 60, elapsedTimeInSeconds % 60);
                            System.out.print("\rOpenSearch domain state: " + state +
" | Time Elapsed: " + formattedTime + " ");
                            System.out.flush();
                            Thread.sleep(1_000);
                        }
                    }
                } catch (InterruptedException e) {
                    Thread.currentThread().interrupt();
                    throw new RuntimeException("Thread was interrupted", e);
                }
            }
        });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ChangeProgress](#)中的。

`CreateDomain`

以下代码示例演示了如何使用 `CreateDomain`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
```

```
    * Creates a new OpenSearch domain asynchronously.
    * @param domainName the name of the new OpenSearch domain to create
    * @return a {@link CompletableFuture} containing the domain ID of the newly
created domain
    */
   public CompletableFuture<String> createNewDomainAsync(String domainName) {
       ClusterConfig clusterConfig = ClusterConfig.builder()
           .dedicatedMasterEnabled(true)
           .dedicatedMasterCount(3)
           .dedicatedMasterType("t2.small.search")
           .instanceType("t2.small.search")
           .instanceCount(5)
           .build();

       EBSOptions ebsOptions = EBSOptions.builder()
           .ebsEnabled(true)
           .volumeSize(10)
           .volumeType(VolumeType.GP2)
           .build();

       NodeToNodeEncryptionOptions encryptionOptions =
NodeToNodeEncryptionOptions.builder()
           .enabled(true)
           .build();

       CreateDomainRequest domainRequest = CreateDomainRequest.builder()
           .domainName(domainName)
           .engineVersion("OpenSearch_1.0")
           .clusterConfig(clusterConfig)
           .ebsOptions(ebsOptions)
           .nodeToNodeEncryptionOptions(encryptionOptions)
           .build();
       logger.info("Sending domain creation request...");
       return getAsyncClient().createDomain(domainRequest)
               .handle( (createResponse, throwable) -> {
                   if (createResponse != null) {
                       logger.info("Domain status is {}",
createResponse.domainStatus().changeProgressDetails().configChangeStatusAsString());
                       logger.info("Domain Id is {}",
createResponse.domainStatus().domainId());
                       return createResponse.domainStatus().domainId();
                   }
                   throw new RuntimeException("Failed to create domain",
throwable);
```

```
                });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateDomain中的。

`DeleteDomain`

以下代码示例演示了如何使用 `DeleteDomain`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Deletes a specific domain asynchronously.
     * @param domainName the name of the domain to be deleted
     * @return a {@link CompletableFuture} that completes when the domain has been
deleted
     * or throws a {@link RuntimeException} if the deletion fails
     */
    public CompletableFuture<DeleteDomainResponse> deleteSpecificDomainAsync(String
domainName) {
        DeleteDomainRequest domainRequest = DeleteDomainRequest.builder()
            .domainName(domainName)
            .build();

        // Delete domain asynchronously
        return getAsyncClient().deleteDomain(domainRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to delete the domain: " +
domainName, exception);
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteDomain中的。

`DescribeDomain`

以下代码示例演示了如何使用 `DescribeDomain`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Describes the specified domain asynchronously.
     *
     * @param domainName the name of the domain to describe
     * @return a {@link CompletableFuture} that completes with the ARN of the domain
     * @throws RuntimeException if the domain description fails
     */
    public CompletableFuture<String> describeDomainAsync(String domainName) {
        DescribeDomainRequest request = DescribeDomainRequest.builder()
            .domainName(domainName)
            .build();

        return getAsyncClient().describeDomain(request)
            .handle((response, exception) -> {  // Handle both response and
exception
                if (exception != null) {
                    throw new RuntimeException("Failed to describe domain",
exception);
                }
                DomainStatus domainStatus = response.domainStatus();
                String endpoint = domainStatus.endpoint();
                String arn = domainStatus.arn();
                String engineVersion = domainStatus.engineVersion();
                logger.info("Domain endpoint is: " + endpoint);
                logger.info("ARN: " + arn);
                System.out.println("Engine version: " + engineVersion);

                return arn;  // Return ARN when successful
```

```
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeDomain中的。

`ListDomainNames`

以下代码示例演示了如何使用 `ListDomainNames`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Asynchronously lists all the domains in the current AWS account.
     * @return a {@link CompletableFuture} that, when completed, contains a list of
{@link DomainInfo} objects representing
     *         the domains in the account.
     * @throws RuntimeException if there was a failure while listing the domains.
     */
    public CompletableFuture<List<DomainInfo>> listAllDomainsAsync() {
        ListDomainNamesRequest namesRequest = ListDomainNamesRequest.builder()
            .engineType("OpenSearch")
            .build();

        return getAsyncClient().listDomainNames(namesRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to list all domains",
exception);
                }
                return response.domainNames();  // Return the list of domain names
on success
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListDomainNames中的。

`ListTags`

以下代码示例演示了如何使用 `ListTags`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Asynchronously adds tags to an Amazon OpenSearch Service domain.
    * @param domainARN the Amazon Resource Name (ARN) of the Amazon OpenSearch
Service domain to add tags to
    * @return a {@link CompletableFuture} that completes when the tags have been
successfully added to the domain,
    * or throws a {@link RuntimeException} if the operation fails
    */
   public CompletableFuture<AddTagsResponse> addDomainTagsAsync(String domainARN) {
       Tag tag1 = Tag.builder()
           .key("service")
           .value("OpenSearch")
           .build();

       Tag tag2 = Tag.builder()
           .key("instances")
           .value("m3.2xlarge")
           .build();

       List<Tag> tagList = new ArrayList<>();
       tagList.add(tag1);
       tagList.add(tag2);

       AddTagsRequest addTagsRequest = AddTagsRequest.builder()
           .arn(domainARN)
           .tagList(tagList)
           .build();
```

```
        return getAsyncClient().addTags(addTagsRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to add tags to the domain: "
+ domainARN, exception);
                } else {
                    logger.info("Added Tags");
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListTags中的。

`UpdateDomainConfig`

以下代码示例演示了如何使用 `UpdateDomainConfig`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Updates the configuration of a specific domain asynchronously.
     * @param domainName the name of the domain to update
     * @return a {@link CompletableFuture} that represents the asynchronous
operation of updating the domain configuration
     */
    public CompletableFuture<UpdateDomainConfigResponse>
updateSpecificDomainAsync(String domainName) {
        ClusterConfig clusterConfig = ClusterConfig.builder()
            .instanceCount(3)
            .build();

        UpdateDomainConfigRequest updateDomainConfigRequest =
UpdateDomainConfigRequest.builder()
```

```
            .domainName(domainName)
            .clusterConfig(clusterConfig)
            .build();

        return getAsyncClient().updateDomainConfig(updateDomainConfigRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Failed to update the domain
configuration", exception);
                }
                // Handle success if needed (e.g., logging or additional actions)
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考UpdateDomainConfig中的。

EventBridge 使用适用于 Java 的 SDK 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

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 基本功能
- 操作
- 场景

开始使用

你好 EventBridge

以下代码示例展示了如何开始使用 EventBridge。

适用于 Java 的 SDK 2.x

> **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。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListEventBuses中的。

基本功能

了解基本功能

以下代码示例展示了如何：

- 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。

适用于 Java 的 SDK 2.x

> **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。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code example performs the following tasks:
 *
 * This Java V2 example performs the following tasks with Amazon EventBridge:
 *
 * 1. Creates an AWS Identity and Access Management (IAM) role to use with
 * Amazon EventBridge.
 * 2. Amazon Simple Storage Service (Amazon S3) bucket with EventBridge events
 * enabled.
 * 3. Creates a rule that triggers when an object is uploaded to Amazon S3.
 * 4. Lists rules on the event bus.
 * 5. Creates a new Amazon Simple Notification Service (Amazon SNS) topic and
 * lets the user subscribe to it.
 * 6. Adds a target to the rule that sends an email to the specified topic.
 * 7. Creates an EventBridge event that sends an email when an Amazon S3 object
 * is created.
 * 8. Lists Targets.
 * 9. Lists the rules for the same target.
 * 10. Triggers the rule by uploading a file to the Amazon S3 bucket.
 * 11. Disables a specific rule.
 * 12. Checks and print the state of the rule.
 * 13. Adds a transform to the rule to change the text of the email.
 * 14. Enables a specific rule.
 * 15. Triggers the updated rule by uploading a file to the Amazon S3 bucket.
 * 16. Updates the rule to be a custom rule pattern.
 * 17. Sending an event to trigger the rule.
 * 18. Cleans up resources.
 *
 */
public class EventbridgeMVP {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException, IOException
 {
        final String usage = """

                Usage:
                    <roleName> <bucketName> <topicName> <eventRuleName>

                Where:
                    roleName - The name of the role to create.
```

```
                    bucketName - The Amazon Simple Storage Service (Amazon S3)
bucket name to create.
                    topicName - The name of the Amazon Simple Notification Service
(Amazon SNS) topic to create.
                    eventRuleName - The Amazon EventBridge rule name to create.
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String polJSON = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                "\"Service\": \"events.amazonaws.com\"" +
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        Scanner sc = new Scanner(System.in);
        String roleName = args[0];
        String bucketName = args[1];
        String topicName = args[2];
        String eventRuleName = args[3];

        Region region = Region.US_EAST_1;
        EventBridgeClient eventBrClient = EventBridgeClient.builder()
                .region(region)
                .build();

        S3Client s3Client = S3Client.builder()
                .region(region)
                .build();

        Region regionGl = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(regionGl)
                .build();

        SnsClient snsClient = SnsClient.builder()
```

```
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon EventBridge example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out
                .println("1. Create an AWS Identity and Access Management (IAM) role
to use with Amazon EventBridge.");
        String roleArn = createIAMRole(iam, roleName, polJSON);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create an S3 bucket with EventBridge events
enabled.");
        if (checkBucket(s3Client, bucketName)) {
            System.out.println("Bucket " + bucketName + " already exists. Ending
this scenario.");
            System.exit(1);
        }

        createBucket(s3Client, bucketName);
        Thread.sleep(3000);
        setBucketNotification(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create a rule that triggers when an object is
uploaded to Amazon S3.");
        Thread.sleep(10000);
        addEventRule(eventBrClient, roleArn, bucketName, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. List rules on the event bus.");
        listRules(eventBrClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Create a new SNS topic for testing and let the user
subscribe to the topic.");
        String topicArn = createSnsTopic(snsClient, topicName);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Add a target to the rule that sends an email to the
specified topic.");
        System.out.println("Enter your email to subscribe to the Amazon SNS
topic:");
        String email = sc.nextLine();
        subEmail(snsClient, topicArn, email);
        System.out.println(
                "Use the link in the email you received to confirm your
subscription. Then, press Enter to continue.");
        sc.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Create an EventBridge event that sends an email when
an Amazon S3 object is created.");
        addSnsEventRule(eventBrClient, eventRuleName, topicArn, topicName,
eventRuleName, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 8. List Targets.");
        listTargets(eventBrClient, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 9. List the rules for the same target.");
        listTargetRules(eventBrClient, topicArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 10. Trigger the rule by uploading a file to the S3
bucket.");
        System.out.println("Press Enter to continue.");
        sc.nextLine();
        uploadTextFiletoS3(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Disable a specific rule.");
        changeRuleState(eventBrClient, eventRuleName, false);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("12. Check and print the state of the rule.");
        checkRule(eventBrClient, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Add a transform to the rule to change the text of
the email.");
        updateSnsEventRule(eventBrClient, topicArn, eventRuleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Enable a specific rule.");
        changeRuleState(eventBrClient, eventRuleName, true);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 15. Trigger the updated rule by uploading a file to the
S3 bucket.");
        System.out.println("Press Enter to continue.");
        sc.nextLine();
        uploadTextFiletoS3(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 16. Update the rule to be a custom rule pattern.");
        updateToCustomRule(eventBrClient, eventRuleName);
        System.out.println("Updated event rule " + eventRuleName + " to use a custom
pattern.");
        updateCustomRuleTargetWithTransform(eventBrClient, topicArn, eventRuleName);
        System.out.println("Updated event target " + topicArn + ".");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Sending an event to trigger the rule. This will
trigger a subscription email.");
        triggerCustomRule(eventBrClient, email);
        System.out.println("Events have been sent. Press Enter to continue.");
        sc.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Clean up resources.");
```

```
        System.out.println("Do you want to clean up resources (y/n)");
        String ans = sc.nextLine();
        if (ans.compareTo("y") == 0) {
            cleanupResources(eventBrClient, snsClient, s3Client, iam, topicArn,
eventRuleName, bucketName, roleName);
        } else {
            System.out.println("The resources will not be cleaned up. ");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The Amazon EventBridge example scenario has successfully
completed.");
        System.out.println(DASHES);
    }

    public static void cleanupResources(EventBridgeClient eventBrClient, SnsClient
snsClient, S3Client s3Client,
            IamClient iam, String topicArn, String eventRuleName, String bucketName,
String roleName) {
        System.out.println("Removing all targets from the event rule.");
        deleteTargetsFromRule(eventBrClient, eventRuleName);
        deleteRuleByName(eventBrClient, eventRuleName);
        deleteSNSTopic(snsClient, topicArn);
        deleteS3Bucket(s3Client, bucketName);
        deleteRole(iam, roleName);
    }

    public static void deleteRole(IamClient iam, String roleName) {
        String policyArn = "arn:aws:iam::aws:policy/AmazonEventBridgeFullAccess";
        DetachRolePolicyRequest policyRequest = DetachRolePolicyRequest.builder()
                .policyArn(policyArn)
                .roleName(roleName)
                .build();

        iam.detachRolePolicy(policyRequest);
        System.out.println("Successfully detached policy " + policyArn + " from role
" + roleName);

        // Delete the role.
        DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                .roleName(roleName)
                .build();
```

```
        iam.deleteRole(roleRequest);
        System.out.println("*** Successfully deleted " + roleName);
    }

    public static void deleteS3Bucket(S3Client s3Client, String bucketName) {
        // Remove all the objects from the S3 bucket.
        ListObjectsRequest listObjects = ListObjectsRequest.builder()
                .bucket(bucketName)
                .build();

        ListObjectsResponse res = s3Client.listObjects(listObjects);
        List<S3Object> objects = res.contents();
        ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();

        for (S3Object myValue : objects) {
            toDelete.add(ObjectIdentifier.builder()
                    .key(myValue.key())
                    .build());
        }

        DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
                .bucket(bucketName)
                .delete(Delete.builder()
                        .objects(toDelete).build())
                .build();

        s3Client.deleteObjects(dor);

        // Delete the S3 bucket.
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();

        s3Client.deleteBucket(deleteBucketRequest);
        System.out.println("You have deleted the bucket and the objects");
    }

    // Delete the SNS topic.
    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                    .topicArn(topicArn)
                    .build();
```

```
            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteRuleByName(EventBridgeClient eventBrClient, String
ruleName) {
        DeleteRuleRequest ruleRequest = DeleteRuleRequest.builder()
                .name(ruleName)
                .build();

        eventBrClient.deleteRule(ruleRequest);
        System.out.println("Successfully deleted the rule");
    }

    public static void deleteTargetsFromRule(EventBridgeClient eventBrClient, String
eventRuleName) {
        // First, get all targets that will be deleted.
        ListTargetsByRuleRequest request = ListTargetsByRuleRequest.builder()
                .rule(eventRuleName)
                .build();

        ListTargetsByRuleResponse response =
eventBrClient.listTargetsByRule(request);
        List<Target> allTargets = response.targets();

        // Get all targets and delete them.
        for (Target myTarget : allTargets) {
            RemoveTargetsRequest removeTargetsRequest =
RemoveTargetsRequest.builder()
                    .rule(eventRuleName)
                    .ids(myTarget.id())
                    .build();

            eventBrClient.removeTargets(removeTargetsRequest);
            System.out.println("Successfully removed the target");
        }
    }
```

```
   public static void triggerCustomRule(EventBridgeClient eventBrClient, String
email) {
       String json = "{" +
               "\"UserEmail\": \"" + email + "\"," +
               "\"Message\": \"This event was generated by example code.\"," +
               "\"UtcTime\": \"Now.\"" +
               "}";

       PutEventsRequestEntry entry = PutEventsRequestEntry.builder()
               .source("ExampleSource")
               .detail(json)
               .detailType("ExampleType")
               .build();

       PutEventsRequest eventsRequest = PutEventsRequest.builder()
               .entries(entry)
               .build();

       eventBrClient.putEvents(eventsRequest);
   }

   public static void updateCustomRuleTargetWithTransform(EventBridgeClient
eventBrClient, String topicArn,
           String ruleName) {
       String targetId = java.util.UUID.randomUUID().toString();
       InputTransformer inputTransformer = InputTransformer.builder()
               .inputTemplate("\"Notification: sample event was received.\"")
               .build();

       Target target = Target.builder()
               .id(targetId)
               .arn(topicArn)
               .inputTransformer(inputTransformer)
               .build();

       try {
           PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
                   .rule(ruleName)
                   .targets(target)
                   .eventBusName(null)
                   .build();

           eventBrClient.putTargets(targetsRequest);
       } catch (EventBridgeException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void updateToCustomRule(EventBridgeClient eventBrClient, String
ruleName) {
        String customEventsPattern = "{" +
                "\"source\": [\"ExampleSource\"]," +
                "\"detail-type\": [\"ExampleType\"]" +
                "}";

        PutRuleRequest request = PutRuleRequest.builder()
                .name(ruleName)
                .description("Custom test rule")
                .eventPattern(customEventsPattern)
                .build();

        eventBrClient.putRule(request);
    }

    // Update an Amazon S3 object created rule with a transform on the target.
    public static void updateSnsEventRule(EventBridgeClient eventBrClient, String
topicArn, String ruleName) {
        String targetId = java.util.UUID.randomUUID().toString();
        Map<String, String> myMap = new HashMap<>();
        myMap.put("bucket", "$.detail.bucket.name");
        myMap.put("time", "$.time");

        InputTransformer inputTransformer = InputTransformer.builder()
                .inputTemplate("\"Notification: an object was uploaded to bucket
<bucket> at <time>.\"")
                .inputPathsMap(myMap)
                .build();

        Target target = Target.builder()
                .id(targetId)
                .arn(topicArn)
                .inputTransformer(inputTransformer)
                .build();

        try {
            PutTargetsRequest targetsRequest = PutTargetsRequest.builder()
                    .rule(ruleName)
```

```
                    .targets(target)
                    .eventBusName(null)
                    .build();

            eventBrClient.putTargets(targetsRequest);

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkRule(EventBridgeClient eventBrClient, String
eventRuleName) {
        try {
            DescribeRuleRequest ruleRequest = DescribeRuleRequest.builder()
                    .name(eventRuleName)
                    .build();

            DescribeRuleResponse response = eventBrClient.describeRule(ruleRequest);
            System.out.println("The state of the rule is " +
response.stateAsString());

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void changeRuleState(EventBridgeClient eventBrClient, String
eventRuleName, Boolean isEnabled) {
        try {
            if (!isEnabled) {
                System.out.println("Disabling the rule: " + eventRuleName);
                DisableRuleRequest ruleRequest = DisableRuleRequest.builder()
                        .name(eventRuleName)
                        .build();

                eventBrClient.disableRule(ruleRequest);
            } else {
                System.out.println("Enabling the rule: " + eventRuleName);
                EnableRuleRequest ruleRequest = EnableRuleRequest.builder()
                        .name(eventRuleName)
                        .build();
```

```
                eventBrClient.enableRule(ruleRequest);
            }

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Create and upload a file to an S3 bucket to trigger an event.
    public static void uploadTextFiletoS3(S3Client s3Client, String bucketName)
throws IOException {
        // Create a unique file name.
        String fileSuffix = new SimpleDateFormat("yyyyMMddHHmmss").format(new
Date());
        String fileName = "TextFile" + fileSuffix + ".txt";

        File myFile = new File(fileName);
        FileWriter fw = new FileWriter(myFile.getAbsoluteFile());
        BufferedWriter bw = new BufferedWriter(fw);
        bw.write("This is a sample file for testing uploads.");
        bw.close();

        try {
            PutObjectRequest putOb = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(fileName)
                    .build();

            s3Client.putObject(putOb, RequestBody.fromFile(myFile));

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listTargetRules(EventBridgeClient eventBrClient, String
topicArn) {
        ListRuleNamesByTargetRequest ruleNamesByTargetRequest =
ListRuleNamesByTargetRequest.builder()
                .targetArn(topicArn)
                .build();
```

```
        ListRuleNamesByTargetResponse response =
eventBrClient.listRuleNamesByTarget(ruleNamesByTargetRequest);
        List<String> rules = response.ruleNames();
        for (String rule : rules) {
            System.out.println("The rule name is " + rule);
        }
    }

    public static void listTargets(EventBridgeClient eventBrClient, String ruleName)
{
        ListTargetsByRuleRequest ruleRequest = ListTargetsByRuleRequest.builder()
                .rule(ruleName)
                .build();

        ListTargetsByRuleResponse res =
eventBrClient.listTargetsByRule(ruleRequest);
        List<Target> targetsList = res.targets();
        for (Target target: targetsList) {
            System.out.println("Target ARN: "+target.arn());
        }
    }

    // Add a rule which triggers an SNS target when a file is uploaded to an S3
    // bucket.
    public static void addSnsEventRule(EventBridgeClient eventBrClient, String
ruleName, String topicArn,
            String topicName, String eventRuleName, String bucketName) {
        String targetID = java.util.UUID.randomUUID().toString();
        Target myTarget = Target.builder()
                .id(targetID)
                .arn(topicArn)
                .build();

        List<Target> targets = new ArrayList<>();
        targets.add(myTarget);
        PutTargetsRequest request = PutTargetsRequest.builder()
                .eventBusName(null)
                .targets(targets)
                .rule(ruleName)
                .build();

        eventBrClient.putTargets(request);
        System.out.println("Added event rule " + eventRuleName + " with Amazon SNS
target " + topicName + " for bucket "
```

```
                + bucketName + ".");
    }

    public static void subEmail(SnsClient snsClient, String topicArn, String email)
 {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("email")
                    .endpoint(email)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() + "\n
\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void listRules(EventBridgeClient eventBrClient) {
        try {
            ListRulesRequest rulesRequest = ListRulesRequest.builder()
                    .eventBusName("default")
                    .limit(10)
                    .build();

            ListRulesResponse response = eventBrClient.listRules(rulesRequest);
            List<Rule> rules = response.rules();
            for (Rule rule : rules) {
                System.out.println("The rule name is : " + rule.name());
                System.out.println("The rule description is : " +
 rule.description());
                System.out.println("The rule state is : " + rule.stateAsString());
            }

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }

    public static String createSnsTopic(SnsClient snsClient, String topicName) {
        String topicPolicy = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Sid\": \"EventBridgePublishTopic\"," +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                "\"Service\": \"events.amazonaws.com\"" +
                "}," +
                "\"Resource\": \"*\"," +
                "\"Action\": \"sns:Publish\"" +
                "}]" +
                "}";

        Map<String, String> topicAttributes = new HashMap<>();
        topicAttributes.put("Policy", topicPolicy);
        CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
                .build();

        CreateTopicResponse response = snsClient.createTopic(topicRequest);
        System.out.println("Added topic " + topicName + " for email
subscriptions.");
        return response.topicArn();
    }

    // Create a new event rule that triggers when an Amazon S3 object is created in
    // a bucket.
    public static void addEventRule(EventBridgeClient eventBrClient, String roleArn,
String bucketName,
            String eventRuleName) {
        String pattern = "{\n" +
                "  \"source\": [\"aws.s3\"],\n" +
                "  \"detail-type\": [\"Object Created\"],\n" +
                "  \"detail\": {\n" +
                "    \"bucket\": {\n" +
                "      \"name\": [\"" + bucketName + "\"]\n" +
                "    }\n" +
                "  }\n" +
                "}";
```

```
        try {
            PutRuleRequest ruleRequest = PutRuleRequest.builder()
                    .description("Created by using the AWS SDK for Java v2")
                    .name(eventRuleName)
                    .eventPattern(pattern)
                    .roleArn(roleArn)
                    .build();

            PutRuleResponse ruleResponse = eventBrClient.putRule(ruleRequest);
            System.out.println("The ARN of the new rule is " +
ruleResponse.ruleArn());

        } catch (EventBridgeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Determine if the S3 bucket exists.
    public static Boolean checkBucket(S3Client s3Client, String bucketName) {
        try {
            HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.headBucket(headBucketRequest);
            return true;
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        return false;
    }

    // Set the S3 bucket notification configuration.
    public static void setBucketNotification(S3Client s3Client, String bucketName) {
        try {
            EventBridgeConfiguration eventBridgeConfiguration =
EventBridgeConfiguration.builder()
                    .build();

            NotificationConfiguration configuration =
NotificationConfiguration.builder()
                    .eventBridgeConfiguration(eventBridgeConfiguration)
                    .build();
```

```
            PutBucketNotificationConfigurationRequest configurationRequest =
PutBucketNotificationConfigurationRequest
                    .builder()
                    .bucket(bucketName)
                    .notificationConfiguration(configuration)
                    .skipDestinationValidation(true)
                    .build();

            s3Client.putBucketNotificationConfiguration(configurationRequest);
            System.out.println("Added bucket " + bucketName + " with EventBridge
events enabled.");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
    public static String createIAMRole(IamClient iam, String rolename, String
 polJSON) {
        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(rolename)
                    .assumeRolePolicyDocument(polJSON)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse response = iam.createRole(request);
            AttachRolePolicyRequest rolePolicyRequest =
 AttachRolePolicyRequest.builder()
                    .roleName(rolename)
                    .policyArn("arn:aws:iam::aws:policy/
AmazonEventBridgeFullAccess")
                    .build();

            iam.attachRolePolicy(rolePolicyRequest);
            return response.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - DeleteRule
  - DescribeRule
  - DisableRule
  - EnableRule
  - ListRuleNamesByTarget
  - ListRules
  - ListTargetsByRule
  - PutEvents
  - PutRule

- PutTargets

操作

`DeleteRule`

以下代码示例演示了如何使用 `DeleteRule`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteRule中的。

`DescribeRule`

以下代码示例演示了如何使用 `DescribeRule`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeRule中的。

`DisableRule`

以下代码示例演示了如何使用 `DisableRule`。

适用于 Java 的 SDK 2.x

> **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。

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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DisableRule中的。

`EnableRule`

以下代码示例演示了如何使用 `EnableRule`。

适用于 Java 的 SDK 2.x

> **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。

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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考EnableRule中的。

`ListRuleNamesByTarget`

以下代码示例演示了如何使用 `ListRuleNamesByTarget`。

适用于 Java 的 SDK 2.x

> **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。

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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ListRuleNamesByTarget](#)中的。

`ListRules`

以下代码示例演示了如何使用 `ListRules`。

适用于 Java 的 SDK 2.x

> **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。

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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListRules中的。

`ListTargetsByRule`

以下代码示例演示了如何使用 `ListTargetsByRule`。

适用于 Java 的 SDK 2.x

> **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。

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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListTargetsByRule中的。

`PutEvents`

以下代码示例演示了如何使用 `PutEvents`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutEvents中的。

`PutRule`

以下代码示例演示了如何使用 `PutRule`。

适用于 Java 的 SDK 2.x

> **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。

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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutRule中的。

`PutTargets`

以下代码示例演示了如何使用 `PutTargets`。

适用于 Java 的 SDK 2.x

> **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。

添加 Amazon SNS 主题，作为规则的目标。

```
   // 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutTargets中的。

`RemoveTargets`

以下代码示例演示了如何使用 `RemoveTargets`。

适用于 Java 的 SDK 2.x

> **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。

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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考RemoveTargets中的。

场景

将事件通知发送至 EventBridge

以下代码示例演示如何允许存储桶向 Amazon SNS 主题和 Amazon SQS 队列发送 S3 事件通知，EventBridge 以及如何将通知路由到 Amazon SNS 主题和 Amazon SQS 队列。

适用于 Java 的 SDK 2.x

> **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。

```
    /** 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《AWS SDK for Java 2.x API Reference》中的以下主题。
  - PutBucketNotificationConfiguration
  - PutRule
  - PutTargets

使用计划的事件调用 Lambda 函数

以下代码示例显示如何创建由 Amazon EventBridge 计划事件调用的 AWS Lambda 函数。

适用于 Java 的 SDK 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。在本示例中，您使用 Lambda Java 运行时 API 创建 Lambda 函数。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

- CloudWatch 日志

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

EventBridge 使用适用于 Java 的 SDK 2.x 的调度器示例

以下代码示例向您展示了如何使用 with S EventBridge cheduler 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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 操作
- 场景

开始使用

你好 S EventBridge cheduler

以下代码示例显示了如何开始使用 EventBridge 调度器。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.services.scheduler.SchedulerAsyncClient;
import software.amazon.awssdk.services.scheduler.model.ListSchedulesRequest;
import software.amazon.awssdk.services.scheduler.model.ScheduleSummary;
import software.amazon.awssdk.services.scheduler.paginators.ListSchedulesPublisher;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.CompletableFuture;

public class HelloScheduler {

    public static void main(String [] args) {
        listSchedulesAsync();
    }

    /**
     * Lists all the schedules available.
     * <p>
     * This method uses the {@link SchedulerAsyncClient} to make an asynchronous
 request to
     * list all the schedules available. The method uses the {@link
 ListSchedulesPublisher}
     * to fetch the schedules in a paginated manner, and then processes the
 responses
     * asynchronously.
     */
    public static void listSchedulesAsync() {
        SchedulerAsyncClient schedulerAsyncClient = SchedulerAsyncClient.create();

        // Build the request to list schedules
        ListSchedulesRequest listSchedulesRequest =
 ListSchedulesRequest.builder().build();

        // Use the paginator to fetch all schedules asynchronously.
        ListSchedulesPublisher paginator =
 schedulerAsyncClient.listSchedulesPaginator(listSchedulesRequest);
        List<ScheduleSummary> results = new ArrayList<>();

        // Subscribe to the paginator to process the response asynchronously
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            response.schedules().forEach(schedule -> {
                results.add(schedule);
                System.out.printf("Schedule: %s%n", schedule.name());
            });
```

```
        });

        // Wait for the asynchronous operation to complete.
        future.join();

        // After all schedules are fetched, print the total count.
        System.out.printf("Total of %d schedule(s) available.%n", results.size());
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ListSchedules](#)中的。

操作

`CreateSchedule`

以下代码示例演示了如何使用 `CreateSchedule`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Creates a new schedule for a target task.
     *
     * @param name                  the name of the schedule
     * @param scheduleExpression    The schedule expression that defines when the
schedule should run.
     * @param scheduleGroupName     the name of the schedule group to which the
schedule belongs
     * @param targetArn             the Amazon Resource Name (ARN) of the target
task
     * @param roleArn               the ARN of the IAM role to be used for the
schedule
     * @param input                 the input data for the target task
```

```
     * @param deleteAfterCompletion whether to delete the schedule after it's
executed
     * @param useFlexibleTimeWindow whether to use a flexible time window for the
schedule execution
     * @return true if the schedule was successfully created, false otherwise
     */
    public CompletableFuture<Boolean> createScheduleAsync(
        String name,
        String scheduleExpression,
        String scheduleGroupName,
        String targetArn,
        String roleArn,
        String input,
        boolean deleteAfterCompletion,
        boolean useFlexibleTimeWindow) {

        int hoursToRun = 1;
        int flexibleTimeWindowMinutes = 10;

        Target target = Target.builder()
            .arn(targetArn)
            .roleArn(roleArn)
            .input(input)
            .build();

        FlexibleTimeWindow flexibleTimeWindow = FlexibleTimeWindow.builder()
            .mode(useFlexibleTimeWindow
                ? FlexibleTimeWindowMode.FLEXIBLE
                : FlexibleTimeWindowMode.OFF)
            .maximumWindowInMinutes(useFlexibleTimeWindow
                ? flexibleTimeWindowMinutes
                : null)
            .build();

        Instant startDate = Instant.now();
        Instant endDate = startDate.plus(Duration.ofHours(hoursToRun));

        CreateScheduleRequest request = CreateScheduleRequest.builder()
            .name(name)
            .scheduleExpression(scheduleExpression)
            .groupName(scheduleGroupName)
            .target(target)
            .actionAfterCompletion(deleteAfterCompletion
                ? ActionAfterCompletion.DELETE
```

```
                : ActionAfterCompletion.NONE)
            .startDate(startDate)
            .endDate(endDate)
            .flexibleTimeWindow(flexibleTimeWindow)
            .build();

        return getAsyncClient().createSchedule(request)
            .thenApply(response -> {
                logger.info("Successfully created schedule {} in schedule group {},
The ARN is {} ", name, scheduleGroupName, response.scheduleArn());
                return true;
            })
            .whenComplete((result, ex) -> {
                if (ex != null) {
                    if (ex instanceof ConflictException) {
                        // Handle ConflictException
                        logger.error("A conflict exception occurred while creating
the schedule: {}", ex.getMessage());
                        throw new CompletionException("A conflict exception occurred
while creating the schedule: " + ex.getMessage(), ex);
                    } else {
                        throw new CompletionException("Error creating schedule: " +
ex.getMessage(), ex);
                    }
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateSchedule中的。

`CreateScheduleGroup`

以下代码示例演示了如何使用 `CreateScheduleGroup`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Creates a new schedule group.
    *
    * @param name the name of the schedule group to be created
    * @return a {@link CompletableFuture} representing the asynchronous operation
of creating the schedule group
    */
   public CompletableFuture<CreateScheduleGroupResponse> createScheduleGroup(String
name) {
       CreateScheduleGroupRequest request = CreateScheduleGroupRequest.builder()
           .name(name)
           .build();

       logger.info("Initiating createScheduleGroup call for group: {}", name);
       CompletableFuture<CreateScheduleGroupResponse> futureResponse =
getAsyncClient().createScheduleGroup(request);
       futureResponse.whenComplete((response, ex) -> {
           if (ex != null) {
               if (ex instanceof CompletionException && ex.getCause() instanceof
ConflictException) {
                   // Rethrow the ConflictException
                   throw (ConflictException) ex.getCause();
               } else {
                   throw new CompletionException("Failed to create schedule group:
" + name, ex);
               }
           } else if (response == null) {
               throw new RuntimeException("Failed to create schedule group:
response was null");
           } else {
               logger.info("Successfully created schedule group '{}': {}", name,
response.scheduleGroupArn());
           }
       });

       return futureResponse;
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateScheduleGroup中的。

DeleteSchedule

以下代码示例演示了如何使用 DeleteSchedule。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Deletes a schedule with the specified name and group name.
    *
    * @param name      the name of the schedule to be deleted
    * @param groupName the group name of the schedule to be deleted
    * @return a {@link CompletableFuture} that, when completed, indicates whether
the schedule was successfully deleted
    * @throws CompletionException if an error occurs while deleting the schedule,
except for the case where the schedule is not found
    */
   public CompletableFuture<Boolean> deleteScheduleAsync(String name, String
groupName) {
       DeleteScheduleRequest request = DeleteScheduleRequest.builder()
           .name(name)
           .groupName(groupName)
           .build();

       CompletableFuture<DeleteScheduleResponse> response =
getAsyncClient().deleteSchedule(request);
       return response.handle((result, ex) -> {
           if (ex != null) {
               if (ex instanceof ResourceNotFoundException) {
                   throw new CompletionException("Resource not found while deleting
schedule with ID: " + name, ex);
               } else {
                   throw new CompletionException("Failed to delete schedule.", ex);
               }
           }
           logger.info("Successfully deleted schedule with name {}.", name);
           return true;
```

```
        });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteSchedule中的。

`DeleteScheduleGroup`

以下代码示例演示了如何使用 `DeleteScheduleGroup`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Deletes the specified schedule group.
     *
     * @param name the name of the schedule group to delete
     * @return a {@link CompletableFuture} that completes when the schedule group
has been deleted
     * @throws CompletionException if an error occurs while deleting the schedule
group
     */
    public CompletableFuture<Void> deleteScheduleGroupAsync(String name) {
        DeleteScheduleGroupRequest request = DeleteScheduleGroupRequest.builder()
            .name(name)
            .build();

        return getAsyncClient().deleteScheduleGroup(request)
            .thenRun(() -> {
                logger.info("Successfully deleted schedule group {}", name);
            })
            .whenComplete((result, ex) -> {
                if (ex != null) {
                    if (ex instanceof ResourceNotFoundException) {
                        throw new CompletionException("The resource was not found: "
+ ex.getMessage(), ex);
```

```
                    } else {
                        throw new CompletionException("Error deleting schedule
group: " + ex.getMessage(), ex);
                    }
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteScheduleGroup中的。

场景

计划的事件

以下代码示例展示了如何：

- 部署包含所需资源的 CloudFormation 堆栈。
- 创建 EventBridge 日程安排组。
- 创建具有灵活时间 EventBridge 范围的一次性日程安排。
- 创建具有指定速率的定期 EventBridge 日程安排。
- 删除 “ EventBridge 日程安排器” 和 “日程组”。
- 清理资源和删除堆栈。

适用于 Java 的 SDK 2.x

> **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。

运行场景。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.scheduler.model.SchedulerException;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
```

```
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Map;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * This Java code example performs the following tasks for the Amazon EventBridge
 Scheduler workflow:
 * <p>
 * 1. Prepare the Application:
 * - Prompt the user for an email address to use for the subscription for the SNS
 topic subscription.
 * - Deploy the Cloud Formation template in resources/cfn_template.yaml for resource
 creation.
 * - Store the outputs of the stack into variables for use in the workflow.
 * - Create a schedule group for all workflow schedules.
 * <p>
 * 2. Create one-time Schedule:
 * - Create a one-time schedule to send an initial event.
 * - Use a Flexible Time Window and set the schedule to delete after completion.
 * - Wait for the user to receive the event email from SNS.
 * <p>
 * 3. Create a time-based schedule:
 * - Prompt the user for how many X times per Y hours a recurring event should be
 scheduled.
 * - Create the scheduled event for X times per hour for Y hours.
 * - Wait for the user to receive the event email from SNS.
 * - Delete the schedule when the user is finished.
 * <p>
 * 4. Clean up:
 * - Prompt the user for y/n answer if they want to destroy the stack and clean up
 all resources.
 * - Delete the schedule group.
 * - Destroy the Cloud Formation stack and wait until the stack has been removed.
 */

public class EventbridgeSchedulerScenario {

    private static final Logger logger =
 LoggerFactory.getLogger(EventbridgeSchedulerScenario.class);
    private static final Scanner scanner = new Scanner(System.in);
    private static String STACK_NAME = "workflow-stack-name";
```

```
    private static final String scheduleGroupName = "schedules-group";

    private static String recurringScheduleName = "";

    private static String oneTimeScheduleName = "";

    private static final EventbridgeSchedulerActions eventbridgeActions = new
EventbridgeSchedulerActions();

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static String roleArn = "";
    public static String snsTopicArn = "";

    public static void main(String[] args) {
        logger.info(DASHES);
        logger.info("Welcome to the Amazon EventBridge Scheduler Workflow.");
        logger.info("""
            Amazon EventBridge Scheduler is a fully managed service that helps you
schedule and execute
            a wide range of tasks and events in the cloud. It's designed to simplify
the process of
            scheduling and managing recurring or one-time events, making it easier
for developers and
            businesses to automate various workflows and processes.

            One of the key features of Amazon EventBridge Scheduler is its ability
to schedule events
            based on a variety of triggers, including time-based schedules, custom
event patterns, or
            even integration with other AWS services. For example, you can use
EventBridge Scheduler
            to schedule a report generation task to run every weekday at 9 AM, or to
trigger a
            Lambda function when a specific Amazon S3 object is created.

            This flexibility allows you to build complex and dynamic event-driven
architectures
            that adapt to your business needs.

            Lets get started...
            """);
        waitForInputToContinue();
        logger.info(DASHES);
```

```
        logger.info(DASHES);
        logger.info("1. Prepare the application.");
        waitForInputToContinue();
        try {
            boolean prepareSuccess = prepareApplication();
            logger.info(DASHES);

            if (prepareSuccess) {
                logger.info("2. Create one-time schedule.");
                logger.info("""
                    A one-time schedule in Amazon EventBridge Scheduler is an event
trigger that allows
                    you to schedule a one-time event to run at a specific date and
time. This is useful for
                    executing a specific task or workflow at a predetermined time,
without the need for recurring
                    or complex scheduling.
                    """);
                waitForInputToContinue();
                createOneTimeSchedule();
                logger.info("Do you want to delete the schedule {} (y/n) ?",
oneTimeScheduleName);
                String ans = scanner.nextLine().trim();
                if (ans.equalsIgnoreCase("y")) {

eventbridgeActions.deleteScheduleAsync(oneTimeScheduleName,scheduleGroupName);
                }
                logger.info(DASHES);

                logger.info("3. Create a recurring schedule.");
                logger.info("""
                    A recurring schedule is a feature that allows you to schedule
and manage the execution
                    of your serverless applications or workloads on a recurring
basis. For example,
                    with EventBridge Scheduler, you can create custom schedules for
your AWS Lambda functions,
                    AWS Step Functions, and other supported event sources, enabling
you to automate tasks and
                    workflows without the need for complex infrastructure
management.
                    """);
                waitForInputToContinue();
```

```
                createRecurringSchedule();
                logger.info("Do you want to delete the schedule {} (y/n) ?",
oneTimeScheduleName);
                String ans2 = scanner.nextLine().trim();
                if (ans2.equalsIgnoreCase("y")) {

eventbridgeActions.deleteScheduleAsync(recurringScheduleName,scheduleGroupName);
                }
                logger.info(DASHES);
            }
        } catch (Exception ex) {
            logger.info("There was a problem with the workflow {}, initiating
cleanup...", ex.getMessage());
            cleanUp();
        }

        logger.info(DASHES);
        logger.info("4. Clean up the resources.");
        logger.info("Do you want to delete these AWS resources (y/n) ?");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            cleanUp();
        } else {
            logger.info("The AWS resources will not be deleted.");
        }
        logger.info("Amazon EventBridge Scheduler workflow completed.");
        logger.info(DASHES);
    }

    /**
     * Cleans up the resources associated with the EventBridge scheduler.
     * If any errors occur during the cleanup process, the corresponding error
messages are logged.
     */
    public static void cleanUp() {
        logger.info("First, delete the schedule group.");
        logger.info("When the schedule group is deleted, schedules that are part of
that group are deleted.");
        waitForInputToContinue();
        try {
            eventbridgeActions.deleteScheduleGroupAsync(scheduleGroupName).join();

        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
```

```
            if (cause instanceof SchedulerException schedulerException) {
                logger.error("Scheduler error occurred: Error message: {}, Error
code {}",
                    schedulerException.getMessage(),
schedulerException.awsErrorDetails().errorCode(), schedulerException);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage());
            }
            return;
        }

        logger.info("Destroy the CloudFormation stack");
        waitForInputToContinue();
        CloudFormationHelper.destroyCloudFormationStack(STACK_NAME);
    }

    /**
     * Prepares the application by creating resources in a CloudFormation stack,
including an SNS topic
     * that will be subscribed to the EventBridge Scheduler events. The user will
need to confirm the subscription
     * in order to receive event emails.
     *
     * @return true if the application preparation was successful, false otherwise
     */
    public static boolean prepareApplication() {
        logger.info("""
            This example creates resources in a CloudFormation stack, including an
SNS topic
            that will be subscribed to the EventBridge Scheduler events.
            You will need to confirm the subscription in order to receive event
emails.
             """);

        String emailAddress = promptUserForEmail();
        logger.info("You entered {}", emailAddress);

        logger.info("Do you want to use a custom Stack name (y/n) ?");
        String ans = scanner.nextLine().trim();
        if (ans.equalsIgnoreCase("y")) {
            String newStackName = scanner.nextLine();
            logger.info("You entered {} for the new stack name", newStackName);
            waitForInputToContinue();
```

```
            STACK_NAME = newStackName;
        }

        logger.info("Get the roleArn and snsTopicArn values using a Cloudformation
template.");
        waitForInputToContinue();
        CloudFormationHelper.deployCloudFormationStack(STACK_NAME, emailAddress);
        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputs(STACK_NAME);
        roleArn = stackOutputs.get("RoleARN");
        snsTopicArn = stackOutputs.get("SNStopicARN");

        logger.info("The roleARN is {}", roleArn);
        logger.info("The snsTopicArn is {}", snsTopicArn);

        try {
            eventbridgeActions.createScheduleGroup(scheduleGroupName).join();
            logger.info("createScheduleGroupAsync completed successfully.");

        } catch (RuntimeException e) {
            logger.error("Error occurred: {} ", e.getMessage());
            return false;
        }
        logger.info("Application preparation complete.");
        return true;
    }

    /**
     * Waits for the user to enter 'c' followed by <ENTER> to continue the program.
     * This method is used to pause the program execution and wait for user input
before
     * proceeding.
     */
    private static void waitForInputToContinue() {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
```

```
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }

    /**
     * Prompts the user to enter an email address and validates the input.
     * If the provided email address is invalid, the method will prompt the user to
try again.
     *
     * @return the valid email address entered by the user
     */
    private static String promptUserForEmail() {
        logger.info("Enter an email address to use for event subscriptions: ");
        String email = scanner.nextLine();
        if (!isValidEmail(email)) {
            logger.info("Invalid email address. Please try again.");
            return promptUserForEmail();
        }
        return email;
    }

    /**
     * Checks if the given email address is valid.
     *
     * @param email the email address to be validated
     * @return {@code true} if the email address is valid, {@code false} otherwise
     */
    private static boolean isValidEmail(String email) {
        try {
            InternetAddress emailAddress = new InternetAddress(email);
            emailAddress.validate();
            return true;

        } catch (AddressException e) {
            return false;
        }
    }

    /**
     * Creates a one-time schedule to send an initial event in 1 minute with a
flexible time window.
     *
```

```
     * @return {@code true} if the schedule was created successfully, {@code false}
 otherwise
     */
    public static Boolean createOneTimeSchedule() {
        oneTimeScheduleName = promptUserForResourceName("Enter a name for the one-
time schedule:");
        logger.info("Creating a one-time schedule named {} to send an initial event
 in 1 minute with a flexible time window...", oneTimeScheduleName);
        LocalDateTime scheduledTime = LocalDateTime.now();
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-
dd'T'HH:mm:ss");

        String scheduleExpression = "at(" + scheduledTime.format(formatter) + ")";
        return eventbridgeActions.createScheduleAsync(
            oneTimeScheduleName,
            scheduleExpression,
            scheduleGroupName,
            snsTopicArn,
            roleArn,
            "One time scheduled event test from schedule",
            true,
            true).join();
    }


    /**
     * Creates a recurring schedule to send events based on a specific time.
     *
     * @return A {@link CompletableFuture} that completes with a boolean value
 indicating the success or failure of the operation.
     */
    public static Boolean createRecurringSchedule() {
        logger.info("Creating a recurring schedule to send events for one hour...");
        recurringScheduleName = promptUserForResourceName("Enter a name for the
 recurring schedule:");

        // Prompt the user for the schedule rate (in minutes).
        int scheduleRateInMinutes = promptUserForInteger("Enter the desired schedule
 rate (in minutes): ");
        String scheduleExpression = "rate(" + scheduleRateInMinutes + " minutes)";
        return eventbridgeActions.createScheduleAsync(
            recurringScheduleName,
            scheduleExpression,
            scheduleGroupName,
```

```
            snsTopicArn,
            roleArn,
            "Recurrent event test from schedule " + recurringScheduleName,
            true,
            true).join();
    }

    /**
     * Prompts the user for a resource name and validates the input.
     *
     * @param prompt the message to display to the user when prompting for the
 resource name
     * @return the valid resource name entered by the user
     */
    private static String promptUserForResourceName(String prompt) {
        logger.info(prompt);
        String resourceName = scanner.nextLine();
        String regex = "[0-9a-zA-Z-_.]+";
        if (!resourceName.matches(regex)) {
            logger.info("Invalid resource name. Please use a name that matches the
 pattern " + regex + ".");
            return promptUserForResourceName(prompt);
        }
        return resourceName;
    }

    /**
     * Prompts the user for an integer input and returns the integer value.
     *
     * @param prompt the message to be displayed to the user when prompting for
 input
     * @return the integer value entered by the user
     */
    private static int promptUserForInteger(String prompt) {
        logger.info(prompt);
        String stringResponse = scanner.nextLine();
        if (stringResponse == null || stringResponse.trim().isEmpty() || !
isInteger(stringResponse)) {
            logger.info("Invalid integer.");
            return promptUserForInteger(prompt);
        }
        return Integer.parseInt(stringResponse);
    }
```

```
    /**
     * Checks if the given string represents a valid integer.
     *
     * @param str the string to be checked
     * @return {@code true} if the string represents a valid integer, {@code false}
 otherwise
     */
    private static boolean isInteger(String str) {
        try {
            Integer.parseInt(str);
            return true;
        } catch (NumberFormatException e) {
            return false;
        }
    }
}
```

适用于服务操作的包装器。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryMode;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.scheduler.SchedulerAsyncClient;
import software.amazon.awssdk.services.scheduler.model.ActionAfterCompletion;
import software.amazon.awssdk.services.scheduler.model.ConflictException;
import software.amazon.awssdk.services.scheduler.model.CreateScheduleGroupRequest;
import software.amazon.awssdk.services.scheduler.model.CreateScheduleGroupResponse;
import software.amazon.awssdk.services.scheduler.model.CreateScheduleRequest;
import software.amazon.awssdk.services.scheduler.model.DeleteScheduleGroupRequest;
import software.amazon.awssdk.services.scheduler.model.DeleteScheduleRequest;
import software.amazon.awssdk.services.scheduler.model.DeleteScheduleResponse;
import software.amazon.awssdk.services.scheduler.model.FlexibleTimeWindow;
import software.amazon.awssdk.services.scheduler.model.FlexibleTimeWindowMode;
import software.amazon.awssdk.services.scheduler.model.ResourceNotFoundException;
import software.amazon.awssdk.services.scheduler.model.Target;

import java.time.Instant;
```

```
import java.util.concurrent.CompletableFuture;
import java.time.Duration;
import java.util.concurrent.CompletionException;

public class EventbridgeSchedulerActions {

    private static SchedulerAsyncClient schedulerClient;
    private static final Logger logger =
 LoggerFactory.getLogger(EventbridgeSchedulerActions.class);

    public static SchedulerAsyncClient getAsyncClient() {
        if (schedulerClient == null) {
            /*
            The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
 version 2,
            and it is designed to provide a high-performance, asynchronous HTTP
 client for interacting with AWS services.
             It uses the Netty framework to handle the underlying network
 communication and the Java NIO API to
             provide a non-blocking, event-driven approach to HTTP requests and
 responses.
             */

            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(50)  // Adjust as needed.
                .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
 timeout.
                .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
                .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
 timeout.
                .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
 individual call attempt timeout.
                .retryStrategy(RetryMode.STANDARD)
                .build();

            schedulerClient = SchedulerAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
```

```
                .build();
        }
        return schedulerClient;
    }


    /**
     * Creates a new schedule group.
     *
     * @param name the name of the schedule group to be created
     * @return a {@link CompletableFuture} representing the asynchronous operation
of creating the schedule group
     */
    public CompletableFuture<CreateScheduleGroupResponse> createScheduleGroup(String
name) {
        CreateScheduleGroupRequest request = CreateScheduleGroupRequest.builder()
            .name(name)
            .build();

        logger.info("Initiating createScheduleGroup call for group: {}", name);
        CompletableFuture<CreateScheduleGroupResponse> futureResponse =
getAsyncClient().createScheduleGroup(request);
        futureResponse.whenComplete((response, ex) -> {
            if (ex != null) {
                if (ex instanceof CompletionException && ex.getCause() instanceof
ConflictException) {
                    // Rethrow the ConflictException
                    throw (ConflictException) ex.getCause();
                } else {
                    throw new CompletionException("Failed to create schedule group:
" + name, ex);
                }
            } else if (response == null) {
                throw new RuntimeException("Failed to create schedule group:
response was null");
            } else {
                logger.info("Successfully created schedule group '{}': {}", name,
response.scheduleGroupArn());
            }
        });

        return futureResponse;
    }
```

```
   /**
    * Creates a new schedule for a target task.
    *
    * @param name                   the name of the schedule
    * @param scheduleExpression    The schedule expression that defines when the
schedule should run.
    * @param scheduleGroupName     the name of the schedule group to which the
schedule belongs
    * @param targetArn             the Amazon Resource Name (ARN) of the target
task
    * @param roleArn               the ARN of the IAM role to be used for the
schedule
    * @param input                 the input data for the target task
    * @param deleteAfterCompletion whether to delete the schedule after it's
executed
    * @param useFlexibleTimeWindow whether to use a flexible time window for the
schedule execution
    * @return true if the schedule was successfully created, false otherwise
    */
   public CompletableFuture<Boolean> createScheduleAsync(
       String name,
       String scheduleExpression,
       String scheduleGroupName,
       String targetArn,
       String roleArn,
       String input,
       boolean deleteAfterCompletion,
       boolean useFlexibleTimeWindow) {

       int hoursToRun = 1;
       int flexibleTimeWindowMinutes = 10;

       Target target = Target.builder()
           .arn(targetArn)
           .roleArn(roleArn)
           .input(input)
           .build();

       FlexibleTimeWindow flexibleTimeWindow = FlexibleTimeWindow.builder()
           .mode(useFlexibleTimeWindow
               ? FlexibleTimeWindowMode.FLEXIBLE
               : FlexibleTimeWindowMode.OFF)
           .maximumWindowInMinutes(useFlexibleTimeWindow
```

```
                ? flexibleTimeWindowMinutes
                : null)
            .build();

        Instant startDate = Instant.now();
        Instant endDate = startDate.plus(Duration.ofHours(hoursToRun));

        CreateScheduleRequest request = CreateScheduleRequest.builder()
            .name(name)
            .scheduleExpression(scheduleExpression)
            .groupName(scheduleGroupName)
            .target(target)
            .actionAfterCompletion(deleteAfterCompletion
                ? ActionAfterCompletion.DELETE
                : ActionAfterCompletion.NONE)
            .startDate(startDate)
            .endDate(endDate)
            .flexibleTimeWindow(flexibleTimeWindow)
            .build();

        return getAsyncClient().createSchedule(request)
            .thenApply(response -> {
                logger.info("Successfully created schedule {} in schedule group {},
The ARN is {} ", name, scheduleGroupName, response.scheduleArn());
                return true;
            })
            .whenComplete((result, ex) -> {
                if (ex != null) {
                    if (ex instanceof ConflictException) {
                        // Handle ConflictException
                        logger.error("A conflict exception occurred while creating
the schedule: {}", ex.getMessage());
                        throw new CompletionException("A conflict exception occurred
while creating the schedule: " + ex.getMessage(), ex);
                    } else {
                        throw new CompletionException("Error creating schedule: " +
ex.getMessage(), ex);
                    }
                }
            });
    }


    /**
```

```
    * Deletes the specified schedule group.
    *
    * @param name the name of the schedule group to delete
    * @return a {@link CompletableFuture} that completes when the schedule group
has been deleted
    * @throws CompletionException if an error occurs while deleting the schedule
group
    */
   public CompletableFuture<Void> deleteScheduleGroupAsync(String name) {
       DeleteScheduleGroupRequest request = DeleteScheduleGroupRequest.builder()
           .name(name)
           .build();

       return getAsyncClient().deleteScheduleGroup(request)
           .thenRun(() -> {
               logger.info("Successfully deleted schedule group {}", name);
           })
           .whenComplete((result, ex) -> {
               if (ex != null) {
                   if (ex instanceof ResourceNotFoundException) {
                       throw new CompletionException("The resource was not found: "
+ ex.getMessage(), ex);
                   } else {
                       throw new CompletionException("Error deleting schedule
group: " + ex.getMessage(), ex);
                   }
               }
           });
   }


   /**
    * Deletes a schedule with the specified name and group name.
    *
    * @param name      the name of the schedule to be deleted
    * @param groupName the group name of the schedule to be deleted
    * @return a {@link CompletableFuture} that, when completed, indicates whether
the schedule was successfully deleted
    * @throws CompletionException if an error occurs while deleting the schedule,
except for the case where the schedule is not found
    */
   public CompletableFuture<Boolean> deleteScheduleAsync(String name, String
groupName) {
       DeleteScheduleRequest request = DeleteScheduleRequest.builder()
```

```
            .name(name)
            .groupName(groupName)
            .build();

        CompletableFuture<DeleteScheduleResponse> response =
 getAsyncClient().deleteSchedule(request);
        return response.handle((result, ex) -> {
            if (ex != null) {
                if (ex instanceof ResourceNotFoundException) {
                    throw new CompletionException("Resource not found while deleting
 schedule with ID: " + name, ex);
                } else {
                    throw new CompletionException("Failed to delete schedule.", ex);
                }
            }
            logger.info("Successfully deleted schedule with name {}.", name);
            return true;
        });
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - CreateSchedule
  - CreateScheduleGroup
  - DeleteSchedule
  - DeleteScheduleGroups

使用 SDK for 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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从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`

以下代码示例演示了如何使用 `CreateDataset`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateDataset中的。

`CreateForecast`

以下代码示例演示了如何使用 `CreateForecast`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateForecast中的。

`DeleteDataset`

以下代码示例演示了如何使用 `DeleteDataset`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteDataset中的。

`DeleteForecast`

以下代码示例演示了如何使用 `DeleteForecast`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[DeleteForecast](#)中的。

DescribeForecast

以下代码示例演示了如何使用 DescribeForecast。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeForecast中的。

`ListDatasetGroups`

以下代码示例演示了如何使用 `ListDatasetGroups`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListDatasetGroups中的。

`ListForecasts`

以下代码示例演示了如何使用 `ListForecasts`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListForecasts中的。

使用适用于 Java 的 SDK 2.x 的 Amazon Glacier 示例

以下代码示例向您展示了如何在 Amazon Glacier 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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从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`

以下代码示例演示了如何使用 `CreateVault`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateVault中的。

`DeleteArchive`

以下代码示例演示了如何使用 `DeleteArchive`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteArchive中的。

`DeleteVault`

以下代码示例演示了如何使用 `DeleteVault`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteVault中的。

`InitiateJob`

以下代码示例演示了如何使用 `InitiateJob`。

适用于 Java 的 SDK 2.x

> **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。

检索文件库清单。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考InitiateJob中的。

`ListVaults`

以下代码示例演示了如何使用 `ListVaults`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListVaults中的。

`UploadArchive`

以下代码示例演示了如何使用 `UploadArchive`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考UploadArchive中的。

AWS Glue 使用适用于 Java 的 SDK 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

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 基本功能
- 操作

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。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListJobs中的。

基本功能

了解基本功能

以下代码示例展示了如何：

- 创建爬网程序，爬取公有 Amazon S3 存储桶并生成包含 CSV 格式的元数据的数据库。
- 列出您的中的数据库和表的相关信息 AWS Glue Data Catalog。
- 创建任务，从 S3 存储桶提取 CSV 数据，转换数据，然后将 JSON 格式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。

适用于 Java 的 SDK 2.x

> **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。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
 *
 * To set up the resources, see this documentation topic:
 *
 * https://docs.aws.amazon.com/glue/latest/ug/tutorial-add-crawler.html
 *
 * This example performs the following tasks:
 *
 * 1. Create a database.
 * 2. Create a crawler.
 * 3. Get a crawler.
 * 4. Start a crawler.
 * 5. Get a database.
 * 6. Get tables.
 * 7. Create a job.
 * 8. Start a job run.
 * 9. List all jobs.
 * 10. Get job runs.
 * 11. Delete a job.
 * 12. Delete a database.
 * 13. Delete a crawler.
 */

public class GlueScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = """

            Usage:
                <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName>
 <scriptLocation> <locationUri> <bucketNameSc>\s

            Where:
                iam - The ARN of the IAM role that has AWS Glue and S3 permissions.
\s
                s3Path - The Amazon Simple Storage Service (Amazon S3) target that
 contains data (for example, s3://<bucket name>/read).
                cron - A cron expression used to specify the schedule  (i.e.,
 cron(15 12 * * ? *).
                dbName - The database name.\s
                crawlerName - The name of the crawler.\s
                jobName - The name you assign to this job definition.
                scriptLocation - The Amazon S3 path to a script that runs a job.
```

```
                locationUri - The location of the database (you can find this file
in resources folder).
                bucketNameSc - The Amazon S3 bucket name used when creating a job
                """;

        if (args.length != 9) {
            System.out.println(usage);
            return;
        }
        Scanner scanner = new Scanner(System.in);
        String iam = args[0];
        String s3Path = args[1];
        String cron = args[2];
        String dbName = args[3];
        String crawlerName = args[4];
        String jobName = args[5];
        String scriptLocation = args[6];
        String locationUri = args[7];
        String bucketNameSc = args[8];

        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
            .region(region)
            .build();
        System.out.println(DASHES);
        System.out.println("Welcome to the AWS Glue scenario.");
        System.out.println("""
            AWS Glue is a fully managed extract, transform, and load (ETL) service
provided by Amazon
            Web Services (AWS). It is designed to simplify the process of building,
running, and maintaining
            ETL pipelines, which are essential for data integration and data
warehousing tasks.

            One of the key features of AWS Glue is its ability to automatically
discover and catalog data
            stored in various sources, such as Amazon S3, Amazon RDS, Amazon
Redshift, and other databases.
            This cataloging process creates a central metadata repository, known as
the AWS Glue Data Catalog,
            which provides a unified view of an organization's data assets. This
metadata can then be used to
            create ETL jobs, which can be scheduled and run on-demand or on a
regular basis.
```

```
            Lets get started.

            """);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create a database.");
        try {
            createDatabase(glueClient, dbName, locationUri);
        } catch (GlueException e) {
            if (e.awsErrorDetails().errorMessage().equals("Database already
exists.")) {
                System.out.println("Database " + dbName + " already exists. Skipping
creation.");
            } else {
                System.err.println(e.awsErrorDetails().errorMessage());
                return;
            }
        }

        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a crawler.");
        try {
            createGlueCrawler(glueClient, iam, s3Path, cron, dbName, crawlerName);
        } catch (GlueException e) {
            if (e.awsErrorDetails().errorMessage().contains("already exists")) {
                System.out.println("Crawler " + crawlerName + " already exists.
Skipping creation.");
            } else {
                System.err.println(e.awsErrorDetails().errorMessage());
                System.exit(1);
            }
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get a crawler.");
        try {
```

```
            getSpecificCrawler(glueClient, crawlerName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Start a crawler.");
        try {
            startSpecificCrawler(glueClient, crawlerName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Get a database.");
        try {
            getSpecificDatabase(glueClient, dbName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Wait 5 min for the tables to become available");
        TimeUnit.MINUTES.sleep(5);
        System.out.println("6. Get tables.");
        String myTableName;
        try {
            myTableName = getGlueTables(glueClient, dbName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("7. Create a job.");
        try {
            createJob(glueClient, jobName, iam, scriptLocation);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Start a Job run.");
        try {
            startJob(glueClient, jobName, dbName, myTableName, bucketNameSc);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. List all jobs.");
        try {
            getAllJobs(glueClient);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get job runs.");
        try {
            getJobRuns(glueClient, jobName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("11. Delete a job.");
        try {
            deleteJob(glueClient, jobName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        System.out.println("*** Wait 5 MIN for the " + crawlerName + " to stop");
        TimeUnit.MINUTES.sleep(5);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Delete a database.");
        try {
            deleteDatabase(glueClient, dbName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Delete a crawler.");
        try {
            deleteSpecificCrawler(glueClient, crawlerName);
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Successfully completed the AWS Glue Scenario");
        System.out.println(DASHES);
    }


    /**
     * Creates a Glue database with the specified name and location URI.
     *
```

```
    * @param glueClient  The Glue client to use for the database creation.
    * @param dbName      The name of the database to create.
    * @param locationUri The location URI for the database.
    */
   public static void createDatabase(GlueClient glueClient, String dbName, String
locationUri) {
       try {
           DatabaseInput input = DatabaseInput.builder()
               .description("Built with the AWS SDK for Java V2")
               .name(dbName)
               .locationUri(locationUri)
               .build();

           CreateDatabaseRequest request = CreateDatabaseRequest.builder()
               .databaseInput(input)
               .build();

           glueClient.createDatabase(request);
           System.out.println(dbName + " was successfully created");

       } catch (GlueException e) {
           throw e;
       }
   }


   /**
    * Creates a new AWS Glue crawler using the AWS Glue Java API.
    *
    * @param glueClient  the AWS Glue client used to interact with the AWS Glue
service
    * @param iam         the IAM role that the crawler will use to access the data
source
    * @param s3Path      the S3 path that the crawler will scan for data
    * @param cron        the cron expression that defines the crawler's schedule
    * @param dbName      the name of the AWS Glue database where the crawler will
store the metadata
    * @param crawlerName the name of the crawler to be created
    */
   public static void createGlueCrawler(GlueClient glueClient,
                                        String iam,
                                        String s3Path,
                                        String cron,
                                        String dbName,
```

```
                                        String crawlerName) {

        try {
            S3Target s3Target = S3Target.builder()
                .path(s3Path)
                .build();

            List<S3Target> targetList = new ArrayList<>();
            targetList.add(s3Target);
            CrawlerTargets targets = CrawlerTargets.builder()
                .s3Targets(targetList)
                .build();

            CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
                .databaseName(dbName)
                .name(crawlerName)
                .description("Created by the AWS Glue Java API")
                .targets(targets)
                .role(iam)
                .schedule(cron)
                .build();

            glueClient.createCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully created");

        } catch (GlueException e) {
            throw e;
        }
    }

    /**
     * Retrieves a specific crawler from the AWS Glue service and waits for it to be
in the "READY" state.
     *
     * @param glueClient  the AWS Glue client used to interact with the Glue service
     * @param crawlerName the name of the crawler to be retrieved
     */
    public static void getSpecificCrawler(GlueClient glueClient, String crawlerName)
throws InterruptedException {
        try {
            GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
                .name(crawlerName)
                .build();
```

```
            boolean ready = false;
            while (!ready) {
                GetCrawlerResponse response = glueClient.getCrawler(crawlerRequest);
                String status = response.crawler().stateAsString();
                if (status.compareTo("READY") == 0) {
                    ready = true;
                }
                Thread.sleep(3000);
            }

            System.out.println("The crawler is now ready");

        } catch (GlueException | InterruptedException e) {
            throw e;
        }
    }

    /**
     * Starts a specific AWS Glue crawler.
     *
     * @param glueClient  the AWS Glue client to use for the crawler operation
     * @param crawlerName the name of the crawler to start
     * @throws GlueException if there is an error starting the crawler
     */
    public static void startSpecificCrawler(GlueClient glueClient, String
crawlerName) {
        try {
            StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            glueClient.startCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully started!");

        } catch (GlueException e) {
            throw e;
        }
    }

    /**
     * Retrieves the specific database from the AWS Glue service.
     *
     * @param glueClient   an instance of the AWS Glue client used to interact with
the service
```

```
    * @param databaseName the name of the database to retrieve
    * @throws GlueException if there is an error retrieving the database from the
AWS Glue service
    */
   public static void getSpecificDatabase(GlueClient glueClient, String
databaseName) {
       try {
           GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
               .name(databaseName)
               .build();

           GetDatabaseResponse response = glueClient.getDatabase(databasesRequest);
           Instant createDate = response.database().createTime();

           // Convert the Instant to readable date.
           DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
               .withLocale(Locale.US)
               .withZone(ZoneId.systemDefault());

           formatter.format(createDate);
           System.out.println("The create date of the database is " + createDate);

       } catch (GlueException e) {
           throw e;
       }
   }


   /**
    * Retrieves the names of the tables in the specified Glue database.
    *
    * @param glueClient the Glue client to use for the operation
    * @param dbName     the name of the Glue database to retrieve the table names
from
    * @return the name of the first table retrieved, or an empty string if no
tables were found
    */
   public static String getGlueTables(GlueClient glueClient, String dbName) {
       String myTableName = "";
       try {
           GetTablesRequest tableRequest = GetTablesRequest.builder()
               .databaseName(dbName)
               .build();
```

```
            GetTablesResponse response = glueClient.getTables(tableRequest);
            List<Table> tables = response.tableList();
            if (tables.isEmpty()) {
                System.out.println("No tables were returned");
            } else {
                for (Table table : tables) {
                    myTableName = table.name();
                    System.out.println("Table name is: " + myTableName);
                }
            }

        } catch (GlueException e) {
            throw e;
        }
        return myTableName;
    }


    /**
     * Starts a job run in AWS Glue.
     *
     * @param glueClient    the AWS Glue client to use for the job run
     * @param jobName       the name of the Glue job to run
     * @param inputDatabase the name of the input database
     * @param inputTable    the name of the input table
     * @param outBucket     the URL of the output S3 bucket
     * @throws GlueException if there is an error starting the job run
     */
    public static void startJob(GlueClient glueClient, String jobName, String
inputDatabase, String inputTable,
                                String outBucket) {
        try {
            Map<String, String> myMap = new HashMap<>();
            myMap.put("--input_database", inputDatabase);
            myMap.put("--input_table", inputTable);
            myMap.put("--output_bucket_url", outBucket);

            StartJobRunRequest runRequest = StartJobRunRequest.builder()
                .workerType(WorkerType.G_1_X)
                .numberOfWorkers(10)
                .arguments(myMap)
                .jobName(jobName)
                .build();
```

```
            StartJobRunResponse response = glueClient.startJobRun(runRequest);
            System.out.println("The request Id of the job is " +
response.responseMetadata().requestId());

        } catch (GlueException e) {
            throw e;
        }
    }


    /**
     * Creates a new AWS Glue job.
     *
     * @param glueClient     the AWS Glue client to use for the operation
     * @param jobName        the name of the job to create
     * @param iam            the IAM role to associate with the job
     * @param scriptLocation the location of the script to be used by the job
     * @throws GlueException if there is an error creating the job
     */
    public static void createJob(GlueClient glueClient, String jobName, String iam,
String scriptLocation) {
        try {
            JobCommand command = JobCommand.builder()
                .pythonVersion("3")
                .name("glueetl")
                .scriptLocation(scriptLocation)
                .build();

            CreateJobRequest jobRequest = CreateJobRequest.builder()
                .description("A Job created by using the AWS SDK for Java V2")
                .glueVersion("2.0")
                .workerType(WorkerType.G_1_X)
                .numberOfWorkers(10)
                .name(jobName)
                .role(iam)
                .command(command)
                .build();

            glueClient.createJob(jobRequest);
            System.out.println(jobName + " was successfully created.");

        } catch (GlueException e) {
            throw e;
```

```
        }
    }


    /**
     * Retrieves and prints information about all the jobs in the Glue data catalog.
     *
     * @param glueClient the Glue client used to interact with the AWS Glue service
     */
    public static void getAllJobs(GlueClient glueClient) {
        try {
            GetJobsRequest jobsRequest = GetJobsRequest.builder()
                .maxResults(10)
                .build();

            GetJobsResponse jobsResponse = glueClient.getJobs(jobsRequest);
            List<Job> jobs = jobsResponse.jobs();
            for (Job job : jobs) {
                System.out.println("Job name is : " + job.name());
                System.out.println("The job worker type is : " +
job.workerType().name());
            }

        } catch (GlueException e) {
            throw e;
        }
    }

    /**
     * Retrieves the job runs for a given Glue job and prints the status of the job
runs.
     *
     * @param glueClient the Glue client used to make API calls
     * @param jobName    the name of the Glue job to retrieve the job runs for
     */
    public static void getJobRuns(GlueClient glueClient, String jobName) {
        try {
            GetJobRunsRequest runsRequest = GetJobRunsRequest.builder()
                .jobName(jobName)
                .maxResults(20)
                .build();

            boolean jobDone = false;
            while (!jobDone) {
```

```
                GetJobRunsResponse response = glueClient.getJobRuns(runsRequest);
                List<JobRun> jobRuns = response.jobRuns();
                for (JobRun jobRun : jobRuns) {
                    String jobState = jobRun.jobRunState().name();
                    if (jobState.compareTo("SUCCEEDED") == 0) {
                        System.out.println(jobName + " has succeeded");
                        jobDone = true;

                    } else if (jobState.compareTo("STOPPED") == 0) {
                        System.out.println("Job run has stopped");
                        jobDone = true;

                    } else if (jobState.compareTo("FAILED") == 0) {
                        System.out.println("Job run has failed");
                        jobDone = true;

                    } else if (jobState.compareTo("TIMEOUT") == 0) {
                        System.out.println("Job run has timed out");
                        jobDone = true;

                    } else {
                        System.out.println("*** Job run state is " +
jobRun.jobRunState().name());
                        System.out.println("Job run Id is " + jobRun.id());
                        System.out.println("The Glue version is " +
jobRun.glueVersion());
                    }
                    TimeUnit.SECONDS.sleep(5);
                }
            }

        } catch (GlueException e) {
            throw e;
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }


    /**
     * Deletes a Glue job.
     *
     * @param glueClient the Glue client to use for the operation
     * @param jobName    the name of the job to be deleted
```

```
     * @throws GlueException if there is an error deleting the job
     */
    public static void deleteJob(GlueClient glueClient, String jobName) {
        try {
            DeleteJobRequest jobRequest = DeleteJobRequest.builder()
                .jobName(jobName)
                .build();

            glueClient.deleteJob(jobRequest);
            System.out.println(jobName + " was successfully deleted");

        } catch (GlueException e) {
            throw e;
        }
    }

    /**
     * Deletes a AWS Glue Database.
     *
     * @param glueClient   An instance of the AWS Glue client used to interact with
the AWS Glue service.
     * @param databaseName The name of the database to be deleted.
     * @throws GlueException If an error occurs while deleting the database.
     */
    public static void deleteDatabase(GlueClient glueClient, String databaseName) {
        try {
            DeleteDatabaseRequest request = DeleteDatabaseRequest.builder()
                .name(databaseName)
                .build();

            glueClient.deleteDatabase(request);
            System.out.println(databaseName + " was successfully deleted");

        } catch (GlueException e) {
            throw e;
        }
    }


    /**
     * Deletes a specific AWS Glue crawler.
     *
     * @param glueClient  the AWS Glue client object
     * @param crawlerName the name of the crawler to be deleted
```

```
     * @throws GlueException if an error occurs during the deletion process
     */
    public static void deleteSpecificCrawler(GlueClient glueClient, String
 crawlerName) {
        try {
            DeleteCrawlerRequest deleteCrawlerRequest =
 DeleteCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            glueClient.deleteCrawler(deleteCrawlerRequest);
            System.out.println(crawlerName + " was deleted");

        } catch (GlueException e) {
            throw e;
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase

- DeleteJob
- DeleteTable
- GetCrawler
- GetDatabase
- GetDatabases
- GetJob
- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

操作

`CreateCrawler`

以下代码示例演示了如何使用 `CreateCrawler`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Creates a new AWS Glue crawler using the AWS Glue Java API.
     *
     * @param glueClient  the AWS Glue client used to interact with the AWS Glue
service
     * @param iam         the IAM role that the crawler will use to access the data
source
     * @param s3Path      the S3 path that the crawler will scan for data
```

```
     * @param cron        the cron expression that defines the crawler's schedule
     * @param dbName      the name of the AWS Glue database where the crawler will
store the metadata
     * @param crawlerName the name of the crawler to be created
     */
    public static void createGlueCrawler(GlueClient glueClient,
                                         String iam,
                                         String s3Path,
                                         String cron,
                                         String dbName,
                                         String crawlerName) {

        try {
            S3Target s3Target = S3Target.builder()
                .path(s3Path)
                .build();

            List<S3Target> targetList = new ArrayList<>();
            targetList.add(s3Target);
            CrawlerTargets targets = CrawlerTargets.builder()
                .s3Targets(targetList)
                .build();

            CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
                .databaseName(dbName)
                .name(crawlerName)
                .description("Created by the AWS Glue Java API")
                .targets(targets)
                .role(iam)
                .schedule(cron)
                .build();

            glueClient.createCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully created");

        } catch (GlueException e) {
            throw e;
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateCrawler中的。

CreateJob

以下代码示例演示了如何使用 CreateJob。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Creates a new AWS Glue job.
    *
    * @param glueClient     the AWS Glue client to use for the operation
    * @param jobName        the name of the job to create
    * @param iam            the IAM role to associate with the job
    * @param scriptLocation the location of the script to be used by the job
    * @throws GlueException if there is an error creating the job
    */
   public static void createJob(GlueClient glueClient, String jobName, String iam,
String scriptLocation) {
       try {
           JobCommand command = JobCommand.builder()
               .pythonVersion("3")
               .name("glueetl")
               .scriptLocation(scriptLocation)
               .build();

           CreateJobRequest jobRequest = CreateJobRequest.builder()
               .description("A Job created by using the AWS SDK for Java V2")
               .glueVersion("2.0")
               .workerType(WorkerType.G_1_X)
               .numberOfWorkers(10)
               .name(jobName)
               .role(iam)
               .command(command)
               .build();

           glueClient.createJob(jobRequest);
           System.out.println(jobName + " was successfully created.");
```

```
        } catch (GlueException e) {
            throw e;
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateJob中的。

`DeleteCrawler`

以下代码示例演示了如何使用 `DeleteCrawler`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Deletes a specific AWS Glue crawler.
     *
     * @param glueClient  the AWS Glue client object
     * @param crawlerName the name of the crawler to be deleted
     * @throws GlueException if an error occurs during the deletion process
     */
    public static void deleteSpecificCrawler(GlueClient glueClient, String
crawlerName) {
        try {
            DeleteCrawlerRequest deleteCrawlerRequest =
DeleteCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            glueClient.deleteCrawler(deleteCrawlerRequest);
            System.out.println(crawlerName + " was deleted");

        } catch (GlueException e) {
```

```
            throw e;
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteCrawler中的。

`DeleteDatabase`

以下代码示例演示了如何使用 `DeleteDatabase`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Deletes a AWS Glue Database.
     *
     * @param glueClient   An instance of the AWS Glue client used to interact with
the AWS Glue service.
     * @param databaseName The name of the database to be deleted.
     * @throws GlueException If an error occurs while deleting the database.
     */
    public static void deleteDatabase(GlueClient glueClient, String databaseName) {
        try {
            DeleteDatabaseRequest request = DeleteDatabaseRequest.builder()
                .name(databaseName)
                .build();

            glueClient.deleteDatabase(request);
            System.out.println(databaseName + " was successfully deleted");

        } catch (GlueException e) {
            throw e;
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteDatabase中的。

`DeleteJob`

以下代码示例演示了如何使用 `DeleteJob`。

适用于 Java 的 SDK 2.x

> **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。

```
/**
 * Deletes a Glue job.
 *
 * @param glueClient the Glue client to use for the operation
 * @param jobName    the name of the job to be deleted
 * @throws GlueException if there is an error deleting the job
 */
public static void deleteJob(GlueClient glueClient, String jobName) {
    try {
        DeleteJobRequest jobRequest = DeleteJobRequest.builder()
            .jobName(jobName)
            .build();

        glueClient.deleteJob(jobRequest);
        System.out.println(jobName + " was successfully deleted");

    } catch (GlueException e) {
        throw e;
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteJob中的。

GetCrawler

以下代码示例演示了如何使用 GetCrawler。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Retrieves a specific crawler from the AWS Glue service and waits for it to be
in the "READY" state.
    *
    * @param glueClient  the AWS Glue client used to interact with the Glue service
    * @param crawlerName the name of the crawler to be retrieved
    */
   public static void getSpecificCrawler(GlueClient glueClient, String crawlerName)
throws InterruptedException {
       try {
           GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
               .name(crawlerName)
               .build();

           boolean ready = false;
           while (!ready) {
               GetCrawlerResponse response = glueClient.getCrawler(crawlerRequest);
               String status = response.crawler().stateAsString();
               if (status.compareTo("READY") == 0) {
                   ready = true;
               }
               Thread.sleep(3000);
           }

           System.out.println("The crawler is now ready");

       } catch (GlueException | InterruptedException e) {
           throw e;
       }
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetCrawler中的。

GetDatabase

以下代码示例演示了如何使用 GetDatabase。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Retrieves the specific database from the AWS Glue service.
    *
    * @param glueClient   an instance of the AWS Glue client used to interact with
the service
    * @param databaseName the name of the database to retrieve
    * @throws GlueException if there is an error retrieving the database from the
AWS Glue service
    */
   public static void getSpecificDatabase(GlueClient glueClient, String
databaseName) {
       try {
           GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
               .name(databaseName)
               .build();

           GetDatabaseResponse response = glueClient.getDatabase(databasesRequest);
           Instant createDate = response.database().createTime();

           // Convert the Instant to readable date.
           DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
               .withLocale(Locale.US)
               .withZone(ZoneId.systemDefault());
```

```
            formatter.format(createDate);
            System.out.println("The create date of the database is " + createDate);

        } catch (GlueException e) {
            throw e;
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetDatabase中的。

`GetJobRuns`

以下代码示例演示了如何使用 `GetJobRuns`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Retrieves the job runs for a given Glue job and prints the status of the job
runs.
     *
     * @param glueClient the Glue client used to make API calls
     * @param jobName    the name of the Glue job to retrieve the job runs for
     */
    public static void getJobRuns(GlueClient glueClient, String jobName) {
        try {
            GetJobRunsRequest runsRequest = GetJobRunsRequest.builder()
                .jobName(jobName)
                .maxResults(20)
                .build();

            boolean jobDone = false;
            while (!jobDone) {
                GetJobRunsResponse response = glueClient.getJobRuns(runsRequest);
                List<JobRun> jobRuns = response.jobRuns();
```

```
                for (JobRun jobRun : jobRuns) {
                    String jobState = jobRun.jobRunState().name();
                    if (jobState.compareTo("SUCCEEDED") == 0) {
                        System.out.println(jobName + " has succeeded");
                        jobDone = true;

                    } else if (jobState.compareTo("STOPPED") == 0) {
                        System.out.println("Job run has stopped");
                        jobDone = true;

                    } else if (jobState.compareTo("FAILED") == 0) {
                        System.out.println("Job run has failed");
                        jobDone = true;

                    } else if (jobState.compareTo("TIMEOUT") == 0) {
                        System.out.println("Job run has timed out");
                        jobDone = true;

                    } else {
                        System.out.println("*** Job run state is " +
jobRun.jobRunState().name());
                        System.out.println("Job run Id is " + jobRun.id());
                        System.out.println("The Glue version is " +
jobRun.glueVersion());
                    }
                    TimeUnit.SECONDS.sleep(5);
                }
            }

        } catch (GlueException e) {
            throw e;
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetJobRuns中的。

GetTables

以下代码示例演示了如何使用 GetTables。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Retrieves the names of the tables in the specified Glue database.
    *
    * @param glueClient the Glue client to use for the operation
    * @param dbName     the name of the Glue database to retrieve the table names
from
    * @return the name of the first table retrieved, or an empty string if no
tables were found
    */
   public static String getGlueTables(GlueClient glueClient, String dbName) {
       String myTableName = "";
       try {
           GetTablesRequest tableRequest = GetTablesRequest.builder()
               .databaseName(dbName)
               .build();

           GetTablesResponse response = glueClient.getTables(tableRequest);
           List<Table> tables = response.tableList();
           if (tables.isEmpty()) {
               System.out.println("No tables were returned");
           } else {
               for (Table table : tables) {
                   myTableName = table.name();
                   System.out.println("Table name is: " + myTableName);
               }
           }

       } catch (GlueException e) {
           throw e;
       }
       return myTableName;
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetTables中的。

`StartCrawler`

以下代码示例演示了如何使用 `StartCrawler`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Starts a specific AWS Glue crawler.
     *
     * @param glueClient  the AWS Glue client to use for the crawler operation
     * @param crawlerName the name of the crawler to start
     * @throws GlueException if there is an error starting the crawler
     */
    public static void startSpecificCrawler(GlueClient glueClient, String
crawlerName) {
        try {
            StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
                .name(crawlerName)
                .build();

            glueClient.startCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully started!");

        } catch (GlueException e) {
            throw e;
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartCrawler中的。

`StartJobRun`

以下代码示例演示了如何使用 `StartJobRun`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Starts a job run in AWS Glue.
    *
    * @param glueClient    the AWS Glue client to use for the job run
    * @param jobName       the name of the Glue job to run
    * @param inputDatabase the name of the input database
    * @param inputTable    the name of the input table
    * @param outBucket     the URL of the output S3 bucket
    * @throws GlueException if there is an error starting the job run
    */
   public static void startJob(GlueClient glueClient, String jobName, String
inputDatabase, String inputTable,
                               String outBucket) {
       try {
           Map<String, String> myMap = new HashMap<>();
           myMap.put("--input_database", inputDatabase);
           myMap.put("--input_table", inputTable);
           myMap.put("--output_bucket_url", outBucket);

           StartJobRunRequest runRequest = StartJobRunRequest.builder()
               .workerType(WorkerType.G_1_X)
               .numberOfWorkers(10)
               .arguments(myMap)
               .jobName(jobName)
               .build();

           StartJobRunResponse response = glueClient.startJobRun(runRequest);
           System.out.println("The request Id of the job is " +
response.responseMetadata().requestId());
```

```
        } catch (GlueException e) {
            throw e;
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartJobRun中的。

HealthImaging 使用适用于 Java 的 SDK 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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从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`

以下代码示例演示了如何使用 `CopyImageSet`。

适用于 Java 的 SDK 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;
        }
    }
```

用于创建可复制的属性的实用程序函数。

```
    /**
     * 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 的详细信息，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示例，了解如何进行设置和运行。

`CreateDatastore`

以下代码示例演示了如何使用 `CreateDatastore`。

适用于 Java 的 SDK 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 的详细信息，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示例，了解如何进行设置和运行。

`DeleteDatastore`

以下代码示例演示了如何使用 `DeleteDatastore`。

适用于 Java 的 SDK 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 的详细信息，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示例，了解如何进行设置和运行。

`DeleteImageSet`

以下代码示例演示了如何使用 `DeleteImageSet`。

适用于 Java 的 SDK 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 的详细信息，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示例，了解如何进行设置和运行。

`GetDICOMImportJob`

以下代码示例演示了如何使用 `GetDICOMImportJob`。

适用于 Java 的 SDK 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考中的 Get DICOMImport J ob。

> **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。

GetDatastore

以下代码示例演示了如何使用 GetDatastore。

适用于 Java 的 SDK 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 的详细信息，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示例，了解如何进行设置和运行。

GetImageFrame

以下代码示例演示了如何使用 GetImageFrame。

适用于 Java 的 SDK 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 的详细信息，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示例，了解如何进行设置和运行。

`GetImageSet`

以下代码示例演示了如何使用 `GetImageSet`。

适用于 Java 的 SDK 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 的详细信息，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示例，了解如何进行设置和运行。

`GetImageSetMetadata`

以下代码示例演示了如何使用 `GetImageSetMetadata`。

适用于 Java 的 SDK 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetImageSetMetadata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示例，了解如何进行设置和运行。

`ListDICOMImportJobs`

以下代码示例演示了如何使用 `ListDICOMImportJobs`。

适用于 Java 的 SDK 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 的详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 “列出DICOMImport作业”。

> **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。

`ListDatastores`

以下代码示例演示了如何使用 `ListDatastores`。

适用于 Java 的 SDK 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 的详细信息，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示例，了解如何进行设置和运行。

`ListImageSetVersions`

以下代码示例演示了如何使用 `ListImageSetVersions`。

适用于 Java 的 SDK 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 的详细信息，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示例，了解如何进行设置和运行。

`ListTagsForResource`

以下代码示例演示了如何使用 `ListTagsForResource`。

适用于 Java 的 SDK 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 的详细信息，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示例，了解如何进行设置和运行。

`SearchImageSets`

以下代码示例演示了如何使用 `SearchImageSets`。

适用于 Java 的 SDK 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;
    }
```

使用案例 #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: 使用 DICOMStudy日期和 DICOMStudy时间的 BETWEEN 运算符。

```
        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();
        }
```

使用案例 #3：使用 createdAt 的 BETWEEN 运算符。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: DICOMSeries InstanceUID 上的 EQUAL 运算符和 updateDat 上的 BETWEEN 运算符，在 updateDat 字段上按照 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 的详细信息，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示例，了解如何进行设置和运行。

`StartDICOMImportJob`

以下代码示例演示了如何使用 `StartDICOMImportJob`。

适用于 Java 的 SDK 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 的详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 “启动 DICOMImport 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示例，了解如何进行设置和运行。

`TagResource`

以下代码示例演示了如何使用 `TagResource`。

适用于 Java 的 SDK 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 的详细信息，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示例，了解如何进行设置和运行。

`UntagResource`

以下代码示例演示了如何使用 `UntagResource`。

适用于 Java 的 SDK 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 的详细信息，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示例，了解如何进行设置和运行。

`UpdateImageSetMetadata`

以下代码示例演示了如何使用 `UpdateImageSetMetadata`。

适用于 Java 的 SDK 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;
        }
    }
```

使用案例 #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()
                                .updatableAttributes(SdkBytes.fromByteBuffer(
                                        ByteBuffer.wrap(insertAttributes
                                                .getBytes(StandardCharsets.UTF_8))))
                                .build())
                        .build();

                updateMedicalImageSetMetadata(medicalImagingClient, datastoreId,
imagesetId,
                        versionid, metadataInsertUpdates, force);
```

使用案例 #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);
```

使用案例 #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);
```

使用案例 #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 的详细信息，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示例，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。

适用于 Java 的 SDK 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《AWS SDK for Java 2.x API Reference》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示例，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。

适用于 Java 的 SDK 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《AWS SDK for Java 2.x API Reference》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示例，了解如何进行设置和运行。

使用 SDK for Java 2.x 的 IAM 示例

以下代码示例向您展示了如何使用 with IAM 来执行操作和实现常见场景。 AWS SDK for Java 2.x

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用

- 基本功能
- 操作
- 场景

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。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListPolicies中的。

基本功能

了解基本功能

以下代码示例展示了如何创建用户并代入角色。

> **Warning**
>
> 为了避免安全风险，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者的联合身份验证，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。

适用于 Java 的 SDK 2.x

> **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。

创建包装 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《AWS SDK for Java 2.x API Reference》中的以下主题。
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole

- CreateUser
- DeleteAccessKey
- DeletePolicy
- DeleteRole
- DeleteUser
- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

操作

`AttachRolePolicy`

以下代码示例演示了如何使用 `AttachRolePolicy`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考AttachRolePolicy中的。

CreateAccessKey

以下代码示例演示了如何使用 CreateAccessKey。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateAccessKey中的。

`CreateAccountAlias`

以下代码示例演示了如何使用 `CreateAccountAlias`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateAccountAlias中的。

`CreatePolicy`

以下代码示例演示了如何使用 `CreatePolicy`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreatePolicy中的。

CreateRole

以下代码示例演示了如何使用 CreateRole。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateRole中的。

`CreateUser`

以下代码示例演示了如何使用 `CreateUser`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateUser中的。

`DeleteAccessKey`

以下代码示例演示了如何使用 `DeleteAccessKey`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteAccessKey中的。

DeleteAccountAlias

以下代码示例演示了如何使用 DeleteAccountAlias。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteAccountAlias中的。

`DeletePolicy`

以下代码示例演示了如何使用 `DeletePolicy`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeletePolicy中的。

`DeleteUser`

以下代码示例演示了如何使用 `DeleteUser`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteUser中的。

DetachRolePolicy

以下代码示例演示了如何使用 DetachRolePolicy。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DetachRolePolicy中的。

`ListAccessKeys`

以下代码示例演示了如何使用 `ListAccessKeys`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListAccessKeys中的。

`ListAccountAliases`

以下代码示例演示了如何使用 `ListAccountAliases`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ListAccountAliases](#)中的。

`ListUsers`

以下代码示例演示了如何使用 `ListUsers`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListUsers中的。

`UpdateAccessKey`

以下代码示例演示了如何使用 `UpdateAccessKey`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考UpdateAccessKey中的。

`UpdateUser`

以下代码示例演示了如何使用 `UpdateUser`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，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 (Amazon EC2) 实例，并将实例数量保持在指定范围内。
- 使用弹性负载均衡处理和分发 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。

适用于 Java 的 SDK 2.x

> **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 wasSuccessul = loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
```

```
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(), StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group for
your default VPC must
                            allow access from this computer. You can either add it
automatically from this
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            System.out.println("Your load balancer is ready. You can access it by
browsing to:");
            System.out.println("\t http://" + elbDnsName);
        } else {
            System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
            System.out.println("manually verifying that your VPC and security group
are configured correctly and that");
```

```
            System.out.println("you can successfully make a GET request to the load
balancer.");
        }

        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
                """
                        This part of the demonstration shows how to toggle
different parts of the system
                        to create situations where the web service fails, and shows
how using a resilient
                        architecture can keep the web service running in spite of
these failures.

                        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                        The table name is contained in a Systems Manager parameter
named self.param_helper.table.
                        To simulate a failure of the recommendation service, let's
set this parameter to name a non-existent table.
                       """);
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");
```

```
        System.out.println(
                """
                        \nNow, sending a GET request to the load balancer endpoint
returns a failure code. But, the service reports as
                        healthy to the load balancer because shallow health checks
don't check for failure of the recommendation service.
                        """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        Instead of failing when the recommendation service fails,
the web service can return a static response.
                        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
                        """);
        paramHelper.put(paramHelper.failureResponse, "static");

        System.out.println("""
                Now, sending a GET request to the load balancer endpoint returns a
static response.
                The service still reports as healthy because health checks are still
shallow.
                """);
        demoChoices(loadBalancer);

        System.out.println("Let's reinstate the recommendation service.");
        paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

        System.out.println("""
                Let's also substitute bad credentials for one of the instances in
the target group so that it can't
                access the DynamoDB recommendation table. We will get an instance id
value.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        AutoScaler autoScaler = new AutoScaler();

        // Create a new instance profile based on badCredsProfileName.
        templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
        String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
```

```
        System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

        String profileAssociationId = autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is " + profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId
                + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

        System.out.println(
                """
                        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
                        depending on which instance is selected by the load
balancer.
                        """);

        demoChoices(loadBalancer);

        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
check tests whether
                the web service can access the DynamoDB table that it depends on for
recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
Scaling instance health.
                This kind of deep health check is not recommended for Auto Scaling
instance health, because it
                risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
                """);

        System.out.println("""
                By implementing deep health checks, the load balancer can detect
when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
                Now, checking target health indicates that the instance with bad
credentials
```

```
                is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy
                instance. Sending a GET request to the load balancer endpoint always
returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
                """);

        demoChoices(loadBalancer);

        System.out.println(
                """
                        Because the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy
                        instance is to terminate it and let the auto scaler start a
new instance to replace it.
                        """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
                Even while the instance is terminating and the new instance is
starting, sending a GET
                request to the web service continues to get a successful
recommendation response because
                the load balancer routes requests to the healthy instances. After
the replacement instance
                starts and reports as healthy, it is included in the load balancing
rotation.
                Note that terminating and replacing an instance typically takes
several minutes, during which time you
                can see the changing health check status until the new instance is
running and healthy.
                """);

        demoChoices(loadBalancer);
        System.out.println(
                "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }
```

```
   public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
       String[] actions = {
               "Send a GET request to the load balancer endpoint.",
               "Check the health of load balancer targets.",
               "Go to the next part of the demo."
       };
       Scanner scanner = new Scanner(System.in);

       while (true) {
           System.out.println("-".repeat(88));
           System.out.println("See the current state of the service by selecting
one of the following choices:");
           for (int i = 0; i < actions.length; i++) {
               System.out.println(i + ": " + actions[i]);
           }

           try {
               System.out.print("\nWhich action would you like to take? ");
               int choice = scanner.nextInt();
               System.out.println("-".repeat(88));

               switch (choice) {
                   case 0 -> {
                       System.out.println("Request:\n");
                       System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                       CloseableHttpClient httpClient =
HttpClients.createDefault();

                       // Create an HTTP GET request to the ELB.
                       HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                       // Execute the request and get the response.
                       HttpResponse response = httpClient.execute(httpGet);
                       int statusCode = response.getStatusLine().getStatusCode();
                       System.out.println("HTTP Status Code: " + statusCode);

                       // Display the JSON response
                       BufferedReader reader = new BufferedReader(
                               new
InputStreamReader(response.getEntity().getContent()));
                       StringBuilder jsonResponse = new StringBuilder();
```

```
                        String line;
                        while ((line = reader.readLine()) != null) {
                            jsonResponse.append(line);
                        }
                        reader.close();

                        // Print the formatted JSON response.
                        System.out.println("Full Response:\n");
                        System.out.println(jsonResponse.toString());

                        // Close the HTTP client.
                        httpClient.close();

                    }
                    case 1 -> {
                        System.out.println("\nChecking the health of load balancer
targets:\n");
                        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
                        for (TargetHealthDescription target : health) {
                            System.out.printf("\tTarget %s on port %d is %s%n",
target.target().id(),
                                    target.target().port(),
target.targetHealth().stateAsString());
                        }
                        System.out.println("""
                                Note that it can take a minute or two for the health
check to update
                                after changes are made.
                                """);
                    }
                    case 2 -> {
                        System.out.println("\nOkay, let's move on.");
                        System.out.println("-".repeat(88));
                        return; // Exit the method when choice is 2
                    }
                    default -> System.out.println("You must choose a value between
0-2. Please select again.");
                }

            } catch (java.util.InputMismatchException e) {
                System.out.println("Invalid input. Please select again.");
                scanner.nextLine(); // Clear the input buffer.
            }
```

```
        }
    }

    public static String readFileAsString(String filePath) throws IOException {
        byte[] bytes = Files.readAllBytes(Paths.get(filePath));
        return new String(bytes);
    }
}
```

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.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
                .instanceIds(instanceId)
                .documentName("AWS-RunShellScript")
                .parameters(Collections.singletonMap("commands",
                        Collections.singletonList("cd / && sudo python3 server.py
80")))
                .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress) {
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(secGroupId)
                .cidrIp(ipAddress)
                .fromPort(Integer.parseInt(port))
                .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
```

```
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                    .builder()
                    .instanceProfileName(profileName)
                    .build();

            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .roleName(roleName)
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(profileRequest);
            DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                    .instanceProfileName(profileName)
                    .build();

            getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
            System.out.println("Deleted instance profile " + profileName);

            DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            // List attached role policies.
            ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                    .listAttachedRolePolicies(role -> role.roleName(roleName));
            List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
            for (AttachedPolicy attachedPolicy : attachedPolicies) {
                DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(attachedPolicy.policyArn())
                        .build();

                getIAMClient().detachRolePolicy(request);
```

```
                System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
            }

            getIAMClient().deleteRole(deleteRoleRequest);
            System.out.println("Instance profile and role deleted.");

        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();


getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network, you
     * must instead specify a prefix list ID. You can also temporarily open the port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     *
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
```

```
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                    .name("group-name")
                    .values("default")
                    .build();

            Filter filter1 = Filter.builder()
                    .name("vpc-id")
                    .values(VPC)
                    .build();

            DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
                    .filters(filter, filter1)
                    .build();

            DescribeSecurityGroupsResponse securityGroupsResponse = getEc2Client()
                    .describeSecurityGroups(securityGroupsRequest);
            String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
            groupInfo.setGroupName(securityGroup);

            for (SecurityGroup secGroup : securityGroupsResponse.securityGroups()) {
                System.out.println("Found security group: " + secGroup.groupId());

                for (IpPermission ipPermission : secGroup.ipPermissions()) {
                    if (ipPermission.fromPort() == port) {
                        System.out.println("Found inbound rule: " + ipPermission);
                        for (IpRange ipRange : ipPermission.ipRanges()) {
                            String cidrIp = ipRange.cidrIp();
                            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                                System.out.println(cidrIp + " is applicable");
                                portIsOpen = true;
                            }
                        }

                        if (!ipPermission.prefixListIds().isEmpty()) {
                            System.out.println("Prefix lList is applicable");
                            portIsOpen = true;
                        }
```

```
                        if (!portIsOpen) {
                            System.out
                                    .println("The inbound rule does not appear to be
open to either this computer's IP,"
                                            + " all IP addresses (0.0.0.0/0), or to
a prefix list ID.");
                        } else {
                            break;
                        }
                    }
                }
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }

        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }

    /*
     * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
     * Scaling group.
     * The target group specifies how the load balancer forward requests to the
     * instances
     * in the group.
     */
    public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
        try {
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                    .autoScalingGroupName(asGroupName)
                    .targetGroupARNs(targetGroupARN)
                    .build();


getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

        // Get availability zones.
        software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
                .launchTemplate(specification)
                .availabilityZones(zones)
                .maxSize(groupSize)
                .minSize(groupSize)
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }

    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();

        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
                .builder()
                .filters(defaultFilter)
                .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
                .name("vpc-id")
                .values(vpcId)
                .build();

        Filter azFilter = Filter.builder()
                .name("availability-zone")
                .values(availabilityZones)
                .build();

        Filter defaultForAZ = Filter.builder()
                .name("default-for-az")
                .values("true")
                .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
                .filters(vpcFilter, azFilter, defaultForAZ)
```

```
                .build();

        DescribeSubnetsResponse response = getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

        DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
                .map(instance -> instance.instanceId())
                .collect(Collectors.toList());

        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }

    // Gets data about the profile associated with an instance.
    public String getInstanceProfile(String instanceId) {
        Filter filter = Filter.builder()
                .name("instance-id")
                .values(instanceId)
                .build();

        DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
                .builder()
                .filters(filter)
                .build();

        DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
                .describeIamInstanceProfileAssociations(associationsRequest);
```

```
        return response.iamInstanceProfileAssociations().get(0).associationId();
    }

    public void deleteRolesPolicies(String policyName, String roleName, String
 InstanceProfile) {
        ListPoliciesRequest listPoliciesRequest =
 ListPoliciesRequest.builder().build();
        ListPoliciesResponse listPoliciesResponse =
 getIAMClient().listPolicies(listPoliciesRequest);
        for (Policy policy : listPoliciesResponse.policies()) {
            if (policy.policyName().equals(policyName)) {
                // List the entities (users, groups, roles) that are attached to the
 policy.

 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
 listEntitiesRequest =
 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                        .builder()
                        .policyArn(policy.arn())
                        .build();
                ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                        .listEntitiesForPolicy(listEntitiesRequest);
                if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                        || !listEntitiesResponse.policyRoles().isEmpty()) {
                    // Detach the policy from any entities it is attached to.
                    DetachRolePolicyRequest detachPolicyRequest =
 DetachRolePolicyRequest.builder()
                            .policyArn(policy.arn())
                            .roleName(roleName) // Specify the name of the IAM role
                            .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
 DeletePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .build();

                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
```

```
                break;
            }
        }

        // List the roles associated with the instance profile
        ListInstanceProfilesForRoleRequest listRolesRequest =
 ListInstanceProfilesForRoleRequest.builder()
                .roleName(roleName)
                .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
 iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
 listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
 RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(InstanceProfile)
                    .roleName(roleName) // Remove the extra dot here
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance profile
 " + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
 DeleteInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .build();

        getIAMClient().deleteInstanceProfile(r ->
 r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

创建一个包装 ELB 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)
                    .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《AWS SDK for Java 2.x API Reference》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

使用 IAM Policy Builder API

以下代码示例展示了如何：

- 使用面向对象的 API 创建 IAM policy。
- 将 IAM Policy Builder API 与 IAM 服务结合使用。

适用于 Java 的 SDK 2.x

> **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。

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:::amzn-s3-
demo-bucket/*")
                                                .addResource("arn:aws:s3:::amzn-s3-
demo-bucket")
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.ARN_NOT_EQUALS)

 .key("aws:PrincipalArn")

 .value("arn:aws:iam::444455556666:user/user-name")))
                                .build();
                return policy.toJson(IamPolicyWriter.builder()
```

```
                                .prettyPrint(true).build());
        }
```

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:::amzn-s3-
demo-bucket/*")
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.STRING_EQUALS)
                                                                .key("s3:x-amz-acl")
                                                                .value("bucket-
owner-full-control")))
                                .build();
                return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true).build());
        }
```

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》https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/feature-iam-policy-builder.html。
- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - CreatePolicy
  - GetPolicy
  - GetPolicyVersion

使用适用于 Java 的 SDK 2.x 的 Amazon Inspector 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 Inspector 配合使用来执行操作和实现常见场景。

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 基本功能
- 操作

开始使用

Hello

以下代码示例展示了如何开始使用。

适用于 Java 的 SDK 2.x

> **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。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloInspector {
    private static final Logger logger =
 LoggerFactory.getLogger(HelloInspector.class);

    public static void main(String[] args) {
        logger.info("Hello Amazon Inspector!");

        try (Inspector2Client inspectorClient = Inspector2Client.builder().build())
 {

            logger.info("Listing member accounts for this Inspector administrator
 account...");
            listMembers(inspectorClient);

            logger.info("The Hello Inspector example completed successfully.");

        } catch (Inspector2Exception e) {
            logger.error("Error: {}", e.getMessage());
            logger.info("Troubleshooting:");
            logger.info("1. Verify AWS credentials are configured");
```

```
            logger.info("2. Check IAM permissions for Inspector2");
            logger.info("3. Ensure Inspector2 is enabled in your account");
            logger.info("4. Verify you're using a supported region");
        }
    }

    /**
     * Lists all member accounts associated with the current Inspector administrator
 account.
     *
     * @param inspectorClient The Inspector2Client used to interact with AWS
 Inspector.
     */
    public static void listMembers(Inspector2Client inspectorClient) {
        try {
            ListMembersRequest request = ListMembersRequest.builder()
                    .maxResults(50) // optional: limit results
                    .build();

            ListMembersResponse response = inspectorClient.listMembers(request);
            List<Member> members = response.members();

            if (members == null || members.isEmpty()) {
                logger.info("No member accounts found for this Inspector
 administrator account.");
                return;
            }

            logger.info("Found {} member account(s):", members.size());
            for (Member member : members) {
                logger.info(" - Account ID: {}, Status: {}",
                        member.accountId(),
                        member.relationshipStatusAsString());
            }

        } catch (Inspector2Exception e) {
            logger.error("Failed to list members: {}",
 e.awsErrorDetails().errorMessage());
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListMembers中的。

基本功能

了解基本功能

以下代码示例展示了如何：

- 查看 Inspector 账户状态。
- 确保已启用 Inspector。
- 分析安全发现。
- 检查扫描覆盖范围。
- 创建调查结果过滤器。
- 列出现有过滤器。
- 查看使用情况和费用。
- 获取覆盖率统计信息。
- 删除过滤器。

适用于 Java 的 SDK 2.x

> **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。

运行一个交互式场景，演示 的功能。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class InspectorScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");
```

```
   private static final Logger logger =
LoggerFactory.getLogger(InspectorScenario.class);
   private static final Scanner scanner = new Scanner(System.in);

   public static void main(String[] args) {
       InspectorActions inspectorActions = new InspectorActions();
       logger.info("Amazon Inspector Basics Scenario");

       logger.info("""
                   Amazon Inspector is a security assessment service provided by
Amazon Web Services (AWS) that helps
                   improve the security and compliance of applications deployed on
AWS. It automatically assesses
                   applications for vulnerabilities or deviations from best
practices. By leveraging Amazon Inspector,
                   users can gain insights into the overall security state of their
application and identify potential
                   security risks.

                   This service operates by conducting both network and host-based
assessments, allowing it to detect a
                   wide range of security issues, including those related to
operating systems, network configurations,
                   and application dependencies.
                   """);

       waitForInputToContinue();

       try {
           runScenario(inspectorActions);

           logger.info("");
           logger.info("Scenario completed successfully!");
           logger.info("");
           logger.info("What you learned:");
           logger.info("  - How to check Inspector account status");
           logger.info("  - How to enable Inspector");
           logger.info("  - How to list and analyze findings");
           logger.info("  - How to check coverage information");
           logger.info("  - How to create and manage filters");
           logger.info("  - How to track usage and costs");
           logger.info("  - How to clean up resources");
           logger.info("");
```

```
        } catch (Exception ex) {
            logger.error("Scenario failed due to unexpected error: {}",
ex.getMessage(), ex);

        } finally {
            scanner.close();
            logger.info("Exiting...");
        }
    }

    /**
     * Runs the Inspector scenario in a step-by-step sequence.
     *
     * All InspectorActions methods are asynchronous and return CompletableFutures.
     * Each step ends with .join(). Any async exception thrown during .join() will
bubble up
     *
     */
    public static void runScenario(InspectorActions actions) {
        String filterArn = null;
        boolean inspectorEnabled = false;

        try {
            // Step 1
            logger.info(DASHES);
            logger.info("Step 1: Checking Inspector account status...");
            String status = actions.getAccountStatusAsync().join();
            logger.info(status);
            waitForInputToContinue();

            // Step 2
            logger.info(DASHES);
            logger.info("Step 2: Enabling Inspector...");
            String message = actions.enableInspectorAsync(null).join();
            logger.info(message);
            inspectorEnabled = true;  // track that Inspector was enabled
            waitForInputToContinue();

            // Step 3
            logger.info(DASHES);
            logger.info("Step 3: Listing LOW severity findings...");

            // Call the service method
```

```
            List<String> allFindings =
actions.listLowSeverityFindingsAsync().join();

            if (!allFindings.isEmpty()) {
                // Only proceed if there are findings
                String lastArn = allFindings.get(allFindings.size() - 1);
                logger.info("Look up details on: {}", lastArn);
                waitForInputToContinue();
                String details = actions.getFindingDetailsAsync(lastArn).join();
                logger.info(details);
            } else {
                logger.info("No LOW severity findings found.");
            }

            waitForInputToContinue();

            // Step 4
            logger.info(DASHES);
            logger.info("Step 4: Listing coverage...");
            String coverage = actions.listCoverageAsync(5).join();
            logger.info(coverage);
            waitForInputToContinue();

            // Step 5
            logger.info(DASHES);
            logger.info("Step 5: Creating filter...");
            String filterName = "suppress-low-" + System.currentTimeMillis();
            filterArn = actions.createLowSeverityFilterAsync(filterName, "Suppress
low severity findings").join();
            logger.info("Created filter: {}", filterArn);
            waitForInputToContinue();

            // Step 6
            logger.info(DASHES);
            logger.info("Step 6: Listing filters...");
            String filters = actions.listFiltersAsync(10).join();
            logger.info(filters);
            waitForInputToContinue();

            // Step 7
            logger.info(DASHES);
            logger.info("Step 7: Usage totals...");
            String usage = actions.listUsageTotalsAsync(null, 10).join();
            logger.info(usage);
```

```
            waitForInputToContinue();

            // Step 8
            logger.info(DASHES);
            logger.info("Step 8: Coverage statistics...");
            String stats = actions.listCoverageStatisticsAsync().join();
            logger.info(stats);
            waitForInputToContinue();

            // Step 9
            logger.info(DASHES);
            logger.info("Step 9: Delete filter?");
            logger.info("Filter ARN: {}", filterArn);
            logger.info("Delete the filter and disable Inspector? (y/n)");

            if (scanner.nextLine().trim().equalsIgnoreCase("y")) {
                actions.deleteFilterAsync(filterArn).join();
                logger.info("Filter deleted.");
                String disableMsg = actions.disableInspectorAsync(null).join();
                logger.info(disableMsg);
                inspectorEnabled = false; // track that Inspector was disabled
            }

            waitForInputToContinue();

        } catch (Exception ex) {
            logger.error("Scenario encountered an error: {}", ex.getMessage(), ex);
            // Rethrow the exception
            throw ex;

        } finally {
            // Cleanup in case of an exception
            if (filterArn != null) {
                try {
                    actions.deleteFilterAsync(filterArn).join();
                    logger.info("Cleanup: Filter deleted.");
                } catch (Exception e) {
                    logger.warn("Failed to delete filter during cleanup: {}",
e.getMessage(), e);
                }
            }

            if (inspectorEnabled) {
                try {
```

```
                    actions.disableInspectorAsync(null).join();
                    logger.info("Cleanup: Inspector disabled.");
                } catch (Exception e) {
                    logger.warn("Failed to disable Inspector during cleanup: {}",
 e.getMessage(), e);
                }
            }
        }
    }

    // Utility Method
    private static void waitForInputToContinue() {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' to continue:");
            String input = scanner.nextLine().trim();
            if (input.equalsIgnoreCase("c")) break;
            logger.info("Invalid input, try again.");
        }
    }
}
```

SDK 方法的封装类。

```
public class InspectorActions {
    private static Inspector2AsyncClient inspectorAsyncClient;
    private static final Logger logger =
 LoggerFactory.getLogger(InspectorActions.class);

    private static Inspector2AsyncClient getAsyncClient() {
        if (inspectorAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                    .maxConcurrency(100)
                    .connectionTimeout(Duration.ofSeconds(60))
                    .readTimeout(Duration.ofSeconds(60))
                    .writeTimeout(Duration.ofSeconds(60))
                    .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
```

```
                    .retryStrategy(RetryMode.STANDARD)
                    .build();

            inspectorAsyncClient = Inspector2AsyncClient.builder()
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
        return inspectorAsyncClient;
    }


    /**
     * Enables AWS Inspector for the provided account(s) and default resource types.
     *
     * @param accountIds Optional list of AWS account IDs.
     */
    public CompletableFuture<String> enableInspectorAsync(List<String> accountIds) {

        // The resource types to enable.
        List<ResourceScanType> resourceTypes = List.of(
                ResourceScanType.EC2,
                ResourceScanType.ECR,
                ResourceScanType.LAMBDA,
                ResourceScanType.LAMBDA_CODE
        );

        // Build the request.
        EnableRequest.Builder requestBuilder = EnableRequest.builder()
                .resourceTypes(resourceTypes);

        if (accountIds != null && !accountIds.isEmpty()) {
            requestBuilder.accountIds(accountIds);
        }

        EnableRequest request = requestBuilder.build();
        return getAsyncClient().enable(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ValidationException) {
                            throw new CompletionException(
                                    "Inspector may already be enabled for this
account: %s".formatted(cause.getMessage()),
```

```
                                    cause
                            );

                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;
                            throw new CompletionException(
                                    "AWS Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to enable Inspector:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    StringBuilder summary = new StringBuilder("Enable results:\n");

                    if (response.accounts() == null ||
response.accounts().isEmpty()) {
                        summary.append("Inspector may already be enabled for all
target accounts.");
                        return summary.toString();
                    }

                    for (Account account : response.accounts()) {
                        String accountId = account.accountId() != null ?
account.accountId() : "Unknown";
                        String status = account.status() != null ?
account.statusAsString() : "Unknown";
                        summary.append(" • Account: ").append(accountId)
                                .append(" # Status: ").append(status).append("\n");
                    }

                    return summary.toString();
                });
    }
```

```
   /**
    * Retrieves and prints the coverage statistics using a paginator.
    */
   public CompletableFuture<String> listCoverageStatisticsAsync() {
       ListCoverageStatisticsRequest request =
ListCoverageStatisticsRequest.builder()
               .build();

       return getAsyncClient().listCoverageStatistics(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();

                       if (cause instanceof ValidationException) {
                           throw new CompletionException(
                                   "Validation error listing coverage statistics:
%s".formatted(cause.getMessage()),
                                   cause
                           );
                       }

                       if (cause instanceof Inspector2Exception) {
                           Inspector2Exception e = (Inspector2Exception) cause;

                           throw new CompletionException(
                                   "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                   e
                           );
                       }

                       throw new CompletionException(
                               "Unexpected error listing coverage statistics:
%s".formatted(exception.getMessage()),
                               exception
                       );
                   }
               })
               .thenApply(response -> {
                   List<Counts> countsList = response.countsByGroup();
                   StringBuilder sb = new StringBuilder();

                   if (countsList == null || countsList.isEmpty()) {
```

```
                        sb.append("No coverage statistics available.\n");
                        return sb.toString();
                    }

                    sb.append("Coverage Statistics:\n");

                    for (Counts c : countsList) {
                        sb.append("  Group: ").append(c.groupKey()).append("\n")
                                .append("    Total Count:
").append(c.count()).append("\n\n");
                    }

                    return sb.toString();
                });
    }


    /**
     * Asynchronously lists Inspector2 usage totals using a paginator.
     *
     * @param accountIds optional list of account IDs
     * @param maxResults maximum results per page
     * @return CompletableFuture completed with formatted summary text
     */
    public CompletableFuture<String> listUsageTotalsAsync(
            List<String> accountIds,
            int maxResults) {

        logger.info("Starting usage totals paginator…");

        ListUsageTotalsRequest.Builder builder = ListUsageTotalsRequest.builder()
                .maxResults(maxResults);

        if (accountIds != null && !accountIds.isEmpty()) {
            builder.accountIds(accountIds);
        }

        ListUsageTotalsRequest request = builder.build();
        ListUsageTotalsPublisher paginator =
getAsyncClient().listUsageTotalsPaginator(request);
        StringBuilder summaryBuilder = new StringBuilder();

        return paginator.subscribe(response -> {
                    if (response.totals() != null && !response.totals().isEmpty()) {
```

```
                        response.totals().forEach(total -> {
                            if (total.usage() != null) {
                                total.usage().forEach(usage -> {
                                    logger.info("Usage: {} = {}",
usage.typeAsString(), usage.total());
                                    summaryBuilder.append(usage.typeAsString())
                                            .append(": ")
                                            .append(usage.total())
                                            .append("\n");
                                });
                            }
                        });
                    } else {
                        logger.info("Page contained no usage totals.");
                    }
                }).thenRun(() -> logger.info("Successfully listed usage totals."))
                .thenApply(v -> {
                    String summary = summaryBuilder.toString();
                    return summary.isEmpty() ? "No usage totals found." : summary;
                }).exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ValidationException ve) {
                        throw new CompletionException(
                                "Validation error listing usage totals:
%s".formatted(ve.getMessage()),
                                ve
                        );
                    }

                    throw new CompletionException("Failed to list usage totals",
cause);
                });
    }



    /**
     * Retrieves the account status using the Inspector2Client.
     */
    public CompletableFuture<String> getAccountStatusAsync() {
        BatchGetAccountStatusRequest request =
BatchGetAccountStatusRequest.builder()
                .accountIds(Collections.emptyList())
```

```
                .build();

        return getAsyncClient().batchGetAccountStatus(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof AccessDeniedException) {
                            throw new CompletionException(
                                    "You do not have sufficient access:
%s".formatted(cause.getMessage()),
                                    cause
                            );

                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;

                            throw new CompletionException(
                                    "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    e
                            );
                        }

                        throw new CompletionException(
                                "Unexpected error getting account status:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {

                    StringBuilder sb = new StringBuilder();
                    List<AccountState> accounts = response.accounts();

                    if (accounts == null || accounts.isEmpty()) {
                        sb.append("No account status returned.\n");
                        return sb.toString();
                    }

                    sb.append("Inspector Account Status:\n");
                    for (AccountState account : accounts) {
```

```
                        String accountId = account.accountId() != null
                                ? account.accountId()
                                : "Unknown";

                        sb.append("  Account ID: ").append(accountId).append("\n");

                        // Overall account state
                        if (account.state() != null && account.state().status() !=
null) {
                            sb.append("  Overall State: ")
                                    .append(account.state().status())
                                    .append("\n");
                        } else {
                            sb.append("  Overall State: Unknown\n");
                        }

                        // Resource state (only status available)
                        ResourceState resources = account.resourceState();
                        if (resources != null) {
                            sb.append("  Resource Status: available\n");
                        }

                        sb.append("\n");
                    }

                    return sb.toString();
                });
    }


    /**
     * Asynchronously lists Inspector2 filters using a paginator.
     *
     * @param maxResults maximum filters per page (nullable)
     * @return CompletableFuture completed with summary text
     */
    public CompletableFuture<String> listFiltersAsync(Integer maxResults) {
        logger.info("Starting async filters paginator…");

        ListFiltersRequest.Builder builder = ListFiltersRequest.builder();
        if (maxResults != null) {
            builder.maxResults(maxResults);
        }
```

```
        ListFiltersRequest request = builder.build();

        // Paginator from SDK
        ListFiltersPublisher paginator =
getAsyncClient().listFiltersPaginator(request);
        StringBuilder collectedFilterIds = new StringBuilder();

        return paginator.subscribe(response -> {
            response.filters().forEach(filter -> {
                logger.info("Filter: " + filter.arn());
                collectedFilterIds.append(filter.arn()).append("\n");
            });
        }).thenApply(v -> {
            String result = collectedFilterIds.toString();
            logger.info("Successfully listed all filters.");
            return result.isEmpty() ? "No filters found." : result;
        }).exceptionally(ex -> {
            Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

            if (cause instanceof ValidationException ve) {
                throw new CompletionException(
                        "Validation error listing filters:
%s".formatted(ve.getMessage()),
                        ve
                );
            }

            throw new RuntimeException("Failed to list filters", ex);
        });
    }


    /**
     * Creates a new LOW severity filter in AWS Inspector2 to suppress findings.
     *
     * @param filterName  the name of the filter to create
     * @param description a descriptive string explaining the purpose of the filter
     * @return a CompletableFuture that completes with the ARN of the created filter
     * @throws CompletionException wraps any validation, Inspector2 service, or
unexpected errors
     */
    public CompletableFuture<String> createLowSeverityFilterAsync(
            String filterName,
```

```
            String description) {

        // Define a filter to match LOW severity findings.
        StringFilter severityFilter = StringFilter.builder()
                .value(Severity.LOW.toString())
                .comparison(StringComparison.EQUALS)
                .build();

        // Create filter criteria.
        FilterCriteria filterCriteria = FilterCriteria.builder()
                .severity(Collections.singletonList(severityFilter))
                .build();

        // Build the filter creation request.
        CreateFilterRequest request = CreateFilterRequest.builder()
                .name(filterName)
                .filterCriteria(filterCriteria)
                .action(FilterAction.SUPPRESS)
                .description(description)
                .build();

        return getAsyncClient().createFilter(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ValidationException ve) {
                            throw new CompletionException(
                                    "Validation error creating filter:
%s".formatted(ve.getMessage()),
                                    ve
                            );
                        }

                        if (cause instanceof Inspector2Exception e) {
                            throw new CompletionException(
                                    "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    e
                            );
                        }

                        // Unexpected async error
```

```
                        throw new CompletionException(
                                "Unexpected error creating filter:
 %s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                // Extract and return the ARN of the created filter.
                .thenApply(CreateFilterResponse::arn);
    }


    /**
     * Lists all AWS Inspector findings of LOW severity asynchronously.
     *
     * @return CompletableFuture containing a List of finding ARNs.
     * Returns an empty list if no LOW severity findings are found.
     */
    public CompletableFuture<ArrayList<String>> listLowSeverityFindingsAsync() {
        logger.info("Starting async LOW severity findings paginator…");

        // Build a filter criteria for LOW severity.
        StringFilter severityFilter = StringFilter.builder()
                .value(Severity.LOW.toString())
                .comparison(StringComparison.EQUALS)
                .build();

        FilterCriteria filterCriteria = FilterCriteria.builder()
                .severity(Collections.singletonList(severityFilter))
                .build();

        // Build the request.
        ListFindingsRequest request = ListFindingsRequest.builder()
                .filterCriteria(filterCriteria)
                .build();

        ListFindingsPublisher paginator =
 getAsyncClient().listFindingsPaginator(request);
        List<String> allArns = Collections.synchronizedList(new ArrayList<>());

        return paginator.subscribe(response -> {
                    if (response.findings() != null && !
response.findings().isEmpty()) {
                        response.findings().forEach(finding -> {
```

```
                            logger.info("Finding ARN: {}", finding.findingArn());
                            allArns.add(finding.findingArn());
                        });
                    } else {
                        logger.info("Page contained no findings.");
                    }
                })
                .thenRun(() -> logger.info("Successfully listed all LOW severity
findings."))
                .thenApply(v -> new ArrayList<>(allArns)) // Return list instead of
a formatted string
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;
                    if (cause instanceof ValidationException ve) {
                        throw new CompletionException(
                                "Validation error listing LOW severity findings:
%s".formatted(ve.getMessage()),
                                ve
                        );
                    }
                    throw new RuntimeException("Failed to list LOW severity
findings", ex);
                });
    }



    /**
     * Lists AWS Inspector2 coverage details for scanned resources using a
paginator.
     *
     * @param maxResults Maximum number of resources to return.
     */
    public CompletableFuture<String> listCoverageAsync(int maxResults) {
        ListCoverageRequest initialRequest = ListCoverageRequest.builder()
                .maxResults(maxResults)
                .build();

        ListCoveragePublisher paginator =
getAsyncClient().listCoveragePaginator(initialRequest);
        StringBuilder summary = new StringBuilder();

        return paginator.subscribe(response -> {
            List<CoveredResource> coveredResources = response.coveredResources();
```

```
            if (coveredResources == null || coveredResources.isEmpty()) {
                summary.append("No coverage information available for this page.
\n");
                return;
            }

            Map<String, List<CoveredResource>> byType = coveredResources.stream()

 .collect(Collectors.groupingBy(CoveredResource::resourceTypeAsString));

            byType.forEach((type, list) ->
                    summary.append("  ").append(type)
                            .append(": ").append(list.size())
                            .append(" resource(s)\n")
            );

            // Include up to 3 sample resources per page
            for (int i = 0; i < Math.min(coveredResources.size(), 3); i++) {
                CoveredResource r = coveredResources.get(i);
                summary.append("  - ").append(r.resourceTypeAsString())
                        .append(": ").append(r.resourceId()).append("\n");
                summary.append("    Scan Type:
 ").append(r.scanTypeAsString()).append("\n");
                if (r.scanStatus() != null) {
                    summary.append("    Status:
 ").append(r.scanStatus().statusCodeAsString()).append("\n");
                }
                if (r.accountId() != null) {
                    summary.append("    Account ID:
 ").append(r.accountId()).append("\n");
                }
                summary.append("\n");
            }

        }).thenApply(v -> {
            if (summary.length() == 0) {
                return "No coverage information found across all pages.";
            } else {
                return "Coverage Information:\n" + summary.toString();
            }
        }).exceptionally(ex -> {
            Throwable cause = ex.getCause();
            if (cause instanceof ValidationException) {
```

```
                throw new CompletionException(
                        "Validation error listing coverage: " + cause.getMessage(),
cause);
            } else if (cause instanceof Inspector2Exception e) {
                throw new CompletionException(
                        "Inspector2 service error: " +
e.awsErrorDetails().errorMessage(), e);
            }
            throw new CompletionException("Unexpected error listing coverage: " +
ex.getMessage(), ex);
        });
    }


    /**
     * Deletes an AWS Inspector2 filter.
     *
     * @param filterARN The ARN of the filter to delete.
     */
    public CompletableFuture<Void> deleteFilterAsync(String filterARN) {
        return getAsyncClient().deleteFilter(
                        DeleteFilterRequest.builder()
                                .arn(filterARN)
                                .build()
                )
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ResourceNotFoundException rnfe) {
                            String msg = "Filter not found for ARN:
%s".formatted(filterARN);
                            logger.warn(msg, rnfe);
                            throw new CompletionException(msg, rnfe);
                        }

                        throw new RuntimeException("Failed to delete the filter: " +
cause, cause);
                    }
                    return null;
                });
    }
```

```
   /**
    * Retrieves detailed information about a specific AWS Inspector2 finding
asynchronously.
    *
    * @param findingArn The ARN of the finding to look up.
    * @return A {@link CompletableFuture} that, when completed, provides a
formatted string
    * containing all available details for the finding.
    * @throws RuntimeException if the async call to Inspector2 fails.
    */
   public CompletableFuture<String> getFindingDetailsAsync(String findingArn) {
       BatchGetFindingDetailsRequest request =
BatchGetFindingDetailsRequest.builder()
               .findingArns(findingArn)
               .build();

       return getAsyncClient().batchGetFindingDetails(request)
               .thenApply(response -> {
                   if (response.findingDetails() == null ||
response.findingDetails().isEmpty()) {
                       return String.format("No details found for ARN: ",
findingArn);
                   }

                   StringBuilder sb = new StringBuilder();
                   response.findingDetails().forEach(detail -> {
                       sb.append("Finding ARN:
").append(detail.findingArn()).append("\n")
                                .append("Risk Score:
").append(detail.riskScore()).append("\n");

                       // ExploitObserved timings
                       if (detail.exploitObserved() != null) {
                           sb.append("Exploit First Seen:
").append(detail.exploitObserved().firstSeen()).append("\n")
                                   .append("Exploit Last Seen:
").append(detail.exploitObserved().lastSeen()).append("\n");
                       }

                       // Reference URLs
                       if (detail.hasReferenceUrls()) {
                           sb.append("Reference URLs:\n");
                           detail.referenceUrls().forEach(url -> sb.append("  •
").append(url).append("\n"));
```

```
                        }

                        // Tools
                        if (detail.hasTools()) {
                            sb.append("Tools:\n");
                            detail.tools().forEach(tool -> sb.append("  •
").append(tool).append("\n"));
                        }

                        // TTPs
                        if (detail.hasTtps()) {
                            sb.append("TTPs:\n");
                            detail.ttps().forEach(ttp -> sb.append("  •
").append(ttp).append("\n"));
                        }

                        // CWEs
                        if (detail.hasCwes()) {
                            sb.append("CWEs:\n");
                            detail.cwes().forEach(cwe -> sb.append("  •
").append(cwe).append("\n"));
                        }

                        // Evidence
                        if (detail.hasEvidences()) {
                            sb.append("Evidence:\n");
                            detail.evidences().forEach(ev -> {
                                sb.append("  - Severity:
").append(ev.severity()).append("\n");

                            });
                        }

                        sb.append("\n");
                    });

                    return sb.toString();
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ResourceNotFoundException rnfe) {
                        return "Finding not found: %s".formatted(findingArn);
                    }
```

```
                    // Fallback for other exceptions
                    throw new RuntimeException("Failed to get finding details for
 ARN: " + findingArn, cause);
                });
    }


    /**
     * Asynchronously disables AWS Inspector for the specified accounts and resource
 types.
     *
     * @param accountIds a {@link List} of AWS account IDs for which to disable
 Inspector;
     *                   may be {@code null} or empty to target the current account
     * @return a {@link CompletableFuture} that, when completed, returns a {@link
 String}
     *         summarizing the disable results for each account
     * @throws CompletionException if the disable operation fails due to validation
 errors,
     *                             service errors, or other exceptions
     * @see <a href="https://docs.aws.amazon.com/inspector/latest/APIReference/
API_Disable.html">
     *      AWS Inspector2 Disable API</a>
     */
    public CompletableFuture<String> disableInspectorAsync(List<String> accountIds)
 {

        // The resource types to disable.
        List<ResourceScanType> resourceTypes = List.of(
                ResourceScanType.EC2,
                ResourceScanType.ECR,
                ResourceScanType.LAMBDA,
                ResourceScanType.LAMBDA_CODE
        );

        // Build the request.
        DisableRequest.Builder requestBuilder = DisableRequest.builder()
                .resourceTypes(resourceTypes);

        if (accountIds != null && !accountIds.isEmpty()) {
            requestBuilder.accountIds(accountIds);
        }
```

```
        DisableRequest request = requestBuilder.build();

        return getAsyncClient().disable(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ValidationException) {
                            throw new CompletionException(
                                    "Inspector may already be disabled for this
account: %s".formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;
                            throw new CompletionException(
                                    "AWS Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to disable Inspector:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    StringBuilder summary = new StringBuilder("Disable results:\n");

                    if (response.accounts() == null ||
response.accounts().isEmpty()) {
                        summary.append("Inspector may already be disabled for all
target accounts.");
                        return summary.toString();
                    }

                    for (Account account : response.accounts()) {
                        String accountId = account.accountId() != null ?
account.accountId() : "Unknown";
```

```
                        String status = account.status() != null ?
 account.statusAsString() : "Unknown";
                        summary.append(" • Account: ").append(accountId)
                                .append(" # Status: ").append(status).append("\n");
                    }

                    return summary.toString();
                });
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - BatchGetAccountStatus
  - BatchGetFindingDetails
  - CreateFilter
  - DeleteFilter
  - 启用
  - ListCoverage
  - ListCoverageStatistics
  - ListFilters
  - ListFindings
  - ListUsageTotals

操作

`BatchGetAccountStatus`

以下代码示例演示了如何使用 `BatchGetAccountStatus`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Retrieves the account status using the Inspector2Client.
    */
   public CompletableFuture<String> getAccountStatusAsync() {
       BatchGetAccountStatusRequest request =
BatchGetAccountStatusRequest.builder()
               .accountIds(Collections.emptyList())
               .build();

       return getAsyncClient().batchGetAccountStatus(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof AccessDeniedException) {
                           throw new CompletionException(
                                   "You do not have sufficient access:
%s".formatted(cause.getMessage()),
                                   cause
                           );

                       }

                       if (cause instanceof Inspector2Exception) {
                           Inspector2Exception e = (Inspector2Exception) cause;

                           throw new CompletionException(
                                   "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                   e
                           );
                       }

                       throw new CompletionException(
                               "Unexpected error getting account status:
%s".formatted(exception.getMessage()),
                               exception
                       );
                   }
               })
               .thenApply(response -> {

                   StringBuilder sb = new StringBuilder();
```

```
                    List<AccountState> accounts = response.accounts();

                    if (accounts == null || accounts.isEmpty()) {
                        sb.append("No account status returned.\n");
                        return sb.toString();
                    }

                    sb.append("Inspector Account Status:\n");
                    for (AccountState account : accounts) {

                        String accountId = account.accountId() != null
                                ? account.accountId()
                                : "Unknown";

                        sb.append("  Account ID: ").append(accountId).append("\n");

                        // Overall account state
                        if (account.state() != null && account.state().status() !=
null) {
                            sb.append("  Overall State: ")
                                    .append(account.state().status())
                                    .append("\n");
                        } else {
                            sb.append("  Overall State: Unknown\n");
                        }

                        // Resource state (only status available)
                        ResourceState resources = account.resourceState();
                        if (resources != null) {
                            sb.append("  Resource Status: available\n");
                        }

                        sb.append("\n");
                    }

                    return sb.toString();
                });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考BatchGetAccountStatus中的。

`BatchGetFindingDetails`

以下代码示例演示了如何使用 `BatchGetFindingDetails`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Retrieves detailed information about a specific AWS Inspector2 finding
asynchronously.
    *
    * @param findingArn The ARN of the finding to look up.
    * @return A {@link CompletableFuture} that, when completed, provides a
formatted string
    * containing all available details for the finding.
    * @throws RuntimeException if the async call to Inspector2 fails.
    */
   public CompletableFuture<String> getFindingDetailsAsync(String findingArn) {
       BatchGetFindingDetailsRequest request =
BatchGetFindingDetailsRequest.builder()
               .findingArns(findingArn)
               .build();

       return getAsyncClient().batchGetFindingDetails(request)
               .thenApply(response -> {
                   if (response.findingDetails() == null ||
response.findingDetails().isEmpty()) {
                       return String.format("No details found for ARN: ",
findingArn);
                   }

                   StringBuilder sb = new StringBuilder();
                   response.findingDetails().forEach(detail -> {
                       sb.append("Finding ARN:
").append(detail.findingArn()).append("\n")
                               .append("Risk Score:
").append(detail.riskScore()).append("\n");
```

```
                        // ExploitObserved timings
                        if (detail.exploitObserved() != null) {
                            sb.append("Exploit First Seen:
").append(detail.exploitObserved().firstSeen()).append("\n")
                                      .append("Exploit Last Seen:
").append(detail.exploitObserved().lastSeen()).append("\n");
                        }

                        // Reference URLs
                        if (detail.hasReferenceUrls()) {
                            sb.append("Reference URLs:\n");
                            detail.referenceUrls().forEach(url -> sb.append("  •
").append(url).append("\n"));
                        }

                        // Tools
                        if (detail.hasTools()) {
                            sb.append("Tools:\n");
                            detail.tools().forEach(tool -> sb.append("  •
").append(tool).append("\n"));
                        }

                        // TTPs
                        if (detail.hasTtps()) {
                            sb.append("TTPs:\n");
                            detail.ttps().forEach(ttp -> sb.append("  •
").append(ttp).append("\n"));
                        }

                        // CWEs
                        if (detail.hasCwes()) {
                            sb.append("CWEs:\n");
                            detail.cwes().forEach(cwe -> sb.append("  •
").append(cwe).append("\n"));
                        }

                        // Evidence
                        if (detail.hasEvidences()) {
                            sb.append("Evidence:\n");
                            detail.evidences().forEach(ev -> {
                                sb.append("  - Severity:
").append(ev.severity()).append("\n");

                            });
```

```
                        }

                        sb.append("\n");
                    });

                    return sb.toString();
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ResourceNotFoundException rnfe) {
                        return "Finding not found: %s".formatted(findingArn);
                    }

                    // Fallback for other exceptions
                    throw new RuntimeException("Failed to get finding details for
ARN: " + findingArn, cause);
                });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考BatchGetFindingDetails中的。

`CreateFilter`

以下代码示例演示了如何使用 `CreateFilter`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Creates a new LOW severity filter in AWS Inspector2 to suppress findings.
     *
     * @param filterName  the name of the filter to create
     * @param description a descriptive string explaining the purpose of the filter
     * @return a CompletableFuture that completes with the ARN of the created filter
```

```
     * @throws CompletionException wraps any validation, Inspector2 service, or
unexpected errors
     */
    public CompletableFuture<String> createLowSeverityFilterAsync(
            String filterName,
            String description) {

        // Define a filter to match LOW severity findings.
        StringFilter severityFilter = StringFilter.builder()
                .value(Severity.LOW.toString())
                .comparison(StringComparison.EQUALS)
                .build();

        // Create filter criteria.
        FilterCriteria filterCriteria = FilterCriteria.builder()
                .severity(Collections.singletonList(severityFilter))
                .build();

        // Build the filter creation request.
        CreateFilterRequest request = CreateFilterRequest.builder()
                .name(filterName)
                .filterCriteria(filterCriteria)
                .action(FilterAction.SUPPRESS)
                .description(description)
                .build();

        return getAsyncClient().createFilter(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ValidationException ve) {
                            throw new CompletionException(
                                    "Validation error creating filter:
%s".formatted(ve.getMessage()),
                                    ve
                            );
                        }

                        if (cause instanceof Inspector2Exception e) {
                            throw new CompletionException(
                                    "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
```

```
                                    e
                            );
                        }

                        // Unexpected async error
                        throw new CompletionException(
                                "Unexpected error creating filter:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                // Extract and return the ARN of the created filter.
                .thenApply(CreateFilterResponse::arn);
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateFilter中的。

`DeleteFilter`

以下代码示例演示了如何使用 `DeleteFilter`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Deletes an AWS Inspector2 filter.
     *
     * @param filterARN The ARN of the filter to delete.
     */
    public CompletableFuture<Void> deleteFilterAsync(String filterARN) {
        return getAsyncClient().deleteFilter(
                        DeleteFilterRequest.builder()
                                .arn(filterARN)
                                .build()
```

```
                )
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ResourceNotFoundException rnfe) {
                            String msg = "Filter not found for ARN:
%s".formatted(filterARN);
                            logger.warn(msg, rnfe);
                            throw new CompletionException(msg, rnfe);
                        }

                        throw new RuntimeException("Failed to delete the filter: " +
cause, cause);
                    }
                    return null;
                });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteFilter中的。

`Disable`

以下代码示例演示了如何使用 `Disable`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Asynchronously disables AWS Inspector for the specified accounts and resource
types.
     *
     * @param accountIds a {@link List} of AWS account IDs for which to disable
Inspector;
     *                   may be {@code null} or empty to target the current account
```

```
     * @return a {@link CompletableFuture} that, when completed, returns a {@link
 String}
     *         summarizing the disable results for each account
     * @throws CompletionException if the disable operation fails due to validation
 errors,
     *                             service errors, or other exceptions
     * @see <a href="https://docs.aws.amazon.com/inspector/latest/APIReference/
API_Disable.html">
     *      AWS Inspector2 Disable API</a>
     */
    public CompletableFuture<String> disableInspectorAsync(List<String> accountIds)
 {

        // The resource types to disable.
        List<ResourceScanType> resourceTypes = List.of(
                ResourceScanType.EC2,
                ResourceScanType.ECR,
                ResourceScanType.LAMBDA,
                ResourceScanType.LAMBDA_CODE
        );

        // Build the request.
        DisableRequest.Builder requestBuilder = DisableRequest.builder()
                .resourceTypes(resourceTypes);

        if (accountIds != null && !accountIds.isEmpty()) {
            requestBuilder.accountIds(accountIds);
        }

        DisableRequest request = requestBuilder.build();

        return getAsyncClient().disable(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ValidationException) {
                            throw new CompletionException(
                                    "Inspector may already be disabled for this
 account: %s".formatted(cause.getMessage()),
                                    cause
                            );
                        }

                        if (cause instanceof Inspector2Exception) {
```

```
                            Inspector2Exception e = (Inspector2Exception) cause;
                            throw new CompletionException(
                                    "AWS Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to disable Inspector:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    StringBuilder summary = new StringBuilder("Disable results:\n");

                    if (response.accounts() == null ||
response.accounts().isEmpty()) {
                        summary.append("Inspector may already be disabled for all
target accounts.");
                        return summary.toString();
                    }

                    for (Account account : response.accounts()) {
                        String accountId = account.accountId() != null ?
account.accountId() : "Unknown";
                        String status = account.status() != null ?
account.statusAsString() : "Unknown";
                        summary.append(" • Account: ").append(accountId)
                                .append(" # Status: ").append(status).append("\n");
                    }

                    return summary.toString();
                });
    }
```

- 有关 API 的详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 “禁用”。

`Enable`

以下代码示例演示了如何使用 `Enable`。

适用于 Java 的 SDK 2.x

> **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。

```
/**
 * Enables AWS Inspector for the provided account(s) and default resource types.
 *
 * @param accountIds Optional list of AWS account IDs.
 */
public CompletableFuture<String> enableInspectorAsync(List<String> accountIds) {

    // The resource types to enable.
    List<ResourceScanType> resourceTypes = List.of(
            ResourceScanType.EC2,
            ResourceScanType.ECR,
            ResourceScanType.LAMBDA,
            ResourceScanType.LAMBDA_CODE
    );

    // Build the request.
    EnableRequest.Builder requestBuilder = EnableRequest.builder()
            .resourceTypes(resourceTypes);

    if (accountIds != null && !accountIds.isEmpty()) {
        requestBuilder.accountIds(accountIds);
    }

    EnableRequest request = requestBuilder.build();
    return getAsyncClient().enable(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ValidationException) {
                        throw new CompletionException(
```

```
                                    "Inspector may already be enabled for this
account: %s".formatted(cause.getMessage()),
                                    cause
                            );

                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;
                            throw new CompletionException(
                                    "AWS Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    cause
                            );
                        }

                        throw new CompletionException(
                                "Failed to enable Inspector:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    StringBuilder summary = new StringBuilder("Enable results:\n");

                    if (response.accounts() == null ||
response.accounts().isEmpty()) {
                        summary.append("Inspector may already be enabled for all
target accounts.");
                        return summary.toString();
                    }

                    for (Account account : response.accounts()) {
                        String accountId = account.accountId() != null ?
account.accountId() : "Unknown";
                        String status = account.status() != null ?
account.statusAsString() : "Unknown";
                        summary.append(" • Account: ").append(accountId)
                                .append(" # Status: ").append(status).append("\n");
                    }

                    return summary.toString();
                });
```

```
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考中的 “启用”。

`ListCoverage`

以下代码示例演示了如何使用 `ListCoverage`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Lists AWS Inspector2 coverage details for scanned resources using a
paginator.
    *
    * @param maxResults Maximum number of resources to return.
    */
   public CompletableFuture<String> listCoverageAsync(int maxResults) {
       ListCoverageRequest initialRequest = ListCoverageRequest.builder()
               .maxResults(maxResults)
               .build();

       ListCoveragePublisher paginator =
getAsyncClient().listCoveragePaginator(initialRequest);
       StringBuilder summary = new StringBuilder();

       return paginator.subscribe(response -> {
           List<CoveredResource> coveredResources = response.coveredResources();

           if (coveredResources == null || coveredResources.isEmpty()) {
               summary.append("No coverage information available for this page.
\n");
               return;
           }
```

```
            Map<String, List<CoveredResource>> byType = coveredResources.stream()

.collect(Collectors.groupingBy(CoveredResource::resourceTypeAsString));

            byType.forEach((type, list) ->
                    summary.append("  ").append(type)
                            .append(": ").append(list.size())
                            .append(" resource(s)\n")
            );

            // Include up to 3 sample resources per page
            for (int i = 0; i < Math.min(coveredResources.size(), 3); i++) {
                CoveredResource r = coveredResources.get(i);
                summary.append("  - ").append(r.resourceTypeAsString())
                        .append(": ").append(r.resourceId()).append("\n");
                summary.append("    Scan Type:
").append(r.scanTypeAsString()).append("\n");
                if (r.scanStatus() != null) {
                    summary.append("    Status:
").append(r.scanStatus().statusCodeAsString()).append("\n");
                }
                if (r.accountId() != null) {
                    summary.append("    Account ID:
").append(r.accountId()).append("\n");
                }
                summary.append("\n");
            }

        }).thenApply(v -> {
            if (summary.length() == 0) {
                return "No coverage information found across all pages.";
            } else {
                return "Coverage Information:\n" + summary.toString();
            }
        }).exceptionally(ex -> {
            Throwable cause = ex.getCause();
            if (cause instanceof ValidationException) {
                throw new CompletionException(
                        "Validation error listing coverage: " + cause.getMessage(),
cause);
            } else if (cause instanceof Inspector2Exception e) {
                throw new CompletionException(
                        "Inspector2 service error: " +
e.awsErrorDetails().errorMessage(), e);
```

```
            }
            throw new CompletionException("Unexpected error listing coverage: " +
ex.getMessage(), ex);
        });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListCoverage中的。

`ListCoverageStatistics`

以下代码示例演示了如何使用 `ListCoverageStatistics`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Retrieves and prints the coverage statistics using a paginator.
     */
    public CompletableFuture<String> listCoverageStatisticsAsync() {
        ListCoverageStatisticsRequest request =
ListCoverageStatisticsRequest.builder()
                .build();

        return getAsyncClient().listCoverageStatistics(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();

                        if (cause instanceof ValidationException) {
                            throw new CompletionException(
                                    "Validation error listing coverage statistics:
%s".formatted(cause.getMessage()),
                                    cause
                            );
```

```
                        }

                        if (cause instanceof Inspector2Exception) {
                            Inspector2Exception e = (Inspector2Exception) cause;

                            throw new CompletionException(
                                    "Inspector2 service error:
%s".formatted(e.awsErrorDetails().errorMessage()),
                                    e
                            );
                        }

                        throw new CompletionException(
                                "Unexpected error listing coverage statistics:
%s".formatted(exception.getMessage()),
                                exception
                        );
                    }
                })
                .thenApply(response -> {
                    List<Counts> countsList = response.countsByGroup();
                    StringBuilder sb = new StringBuilder();

                    if (countsList == null || countsList.isEmpty()) {
                        sb.append("No coverage statistics available.\n");
                        return sb.toString();
                    }

                    sb.append("Coverage Statistics:\n");

                    for (Counts c : countsList) {
                        sb.append("  Group: ").append(c.groupKey()).append("\n")
                                .append("    Total Count:
").append(c.count()).append("\n\n");
                    }

                    return sb.toString();
                });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ListCoverageStatistics](#)中的。

`ListFilters`

以下代码示例演示了如何使用 `ListFilters`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Asynchronously lists Inspector2 filters using a paginator.
     *
     * @param maxResults maximum filters per page (nullable)
     * @return CompletableFuture completed with summary text
     */
    public CompletableFuture<String> listFiltersAsync(Integer maxResults) {
        logger.info("Starting async filters paginator…");

        ListFiltersRequest.Builder builder = ListFiltersRequest.builder();
        if (maxResults != null) {
            builder.maxResults(maxResults);
        }

        ListFiltersRequest request = builder.build();

        // Paginator from SDK
        ListFiltersPublisher paginator =
getAsyncClient().listFiltersPaginator(request);
        StringBuilder collectedFilterIds = new StringBuilder();

        return paginator.subscribe(response -> {
            response.filters().forEach(filter -> {
                logger.info("Filter: " + filter.arn());
                collectedFilterIds.append(filter.arn()).append("\n");
            });
        }).thenApply(v -> {
            String result = collectedFilterIds.toString();
            logger.info("Successfully listed all filters.");
            return result.isEmpty() ? "No filters found." : result;
```

```
        }).exceptionally(ex -> {
            Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

            if (cause instanceof ValidationException ve) {
                throw new CompletionException(
                        "Validation error listing filters:
%s".formatted(ve.getMessage()),
                        ve
                );
            }

            throw new RuntimeException("Failed to list filters", ex);
        });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListFilters中的。

`ListFindings`

以下代码示例演示了如何使用 `ListFindings`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Lists all AWS Inspector findings of LOW severity asynchronously.
     *
     * @return CompletableFuture containing a List of finding ARNs.
     * Returns an empty list if no LOW severity findings are found.
     */
    public CompletableFuture<ArrayList<String>> listLowSeverityFindingsAsync() {
        logger.info("Starting async LOW severity findings paginator…");

        // Build a filter criteria for LOW severity.
        StringFilter severityFilter = StringFilter.builder()
```

```
                .value(Severity.LOW.toString())
                .comparison(StringComparison.EQUALS)
                .build();

        FilterCriteria filterCriteria = FilterCriteria.builder()
                .severity(Collections.singletonList(severityFilter))
                .build();

        // Build the request.
        ListFindingsRequest request = ListFindingsRequest.builder()
                .filterCriteria(filterCriteria)
                .build();

        ListFindingsPublisher paginator =
 getAsyncClient().listFindingsPaginator(request);
        List<String> allArns = Collections.synchronizedList(new ArrayList<>());

        return paginator.subscribe(response -> {
                    if (response.findings() != null && !
response.findings().isEmpty()) {
                        response.findings().forEach(finding -> {
                            logger.info("Finding ARN: {}", finding.findingArn());
                            allArns.add(finding.findingArn());
                        });
                    } else {
                        logger.info("Page contained no findings.");
                    }
                })
                .thenRun(() -> logger.info("Successfully listed all LOW severity
 findings."))
                .thenApply(v -> new ArrayList<>(allArns)) // Return list instead of
 a formatted string
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;
                    if (cause instanceof ValidationException ve) {
                        throw new CompletionException(
                                "Validation error listing LOW severity findings:
 %s".formatted(ve.getMessage()),
                                ve
                        );
                    }
                    throw new RuntimeException("Failed to list LOW severity
 findings", ex);
                });
```

```
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListFindings中的。

`ListUsageTotals`

以下代码示例演示了如何使用 `ListUsageTotals`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Asynchronously lists Inspector2 usage totals using a paginator.
     *
     * @param accountIds optional list of account IDs
     * @param maxResults maximum results per page
     * @return CompletableFuture completed with formatted summary text
     */
    public CompletableFuture<String> listUsageTotalsAsync(
            List<String> accountIds,
            int maxResults) {

        logger.info("Starting usage totals paginator…");

        ListUsageTotalsRequest.Builder builder = ListUsageTotalsRequest.builder()
                .maxResults(maxResults);

        if (accountIds != null && !accountIds.isEmpty()) {
            builder.accountIds(accountIds);
        }

        ListUsageTotalsRequest request = builder.build();
        ListUsageTotalsPublisher paginator =
getAsyncClient().listUsageTotalsPaginator(request);
```

```
        StringBuilder summaryBuilder = new StringBuilder();

        return paginator.subscribe(response -> {
                    if (response.totals() != null && !response.totals().isEmpty()) {
                        response.totals().forEach(total -> {
                            if (total.usage() != null) {
                                total.usage().forEach(usage -> {
                                    logger.info("Usage: {} = {}",
usage.typeAsString(), usage.total());
                                    summaryBuilder.append(usage.typeAsString())
                                            .append(": ")
                                            .append(usage.total())
                                            .append("\n");
                                });
                            }
                        });
                    } else {
                        logger.info("Page contained no usage totals.");
                    }
                }).thenRun(() -> logger.info("Successfully listed usage totals."))
                .thenApply(v -> {
                    String summary = summaryBuilder.toString();
                    return summary.isEmpty() ? "No usage totals found." : summary;
                }).exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ValidationException ve) {
                        throw new CompletionException(
                                "Validation error listing usage totals:
%s".formatted(ve.getMessage()),
                                ve
                        );
                    }

                    throw new CompletionException("Failed to list usage totals",
cause);
                });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListUsageTotals中的。

AWS IoT 使用适用于 Java 的 SDK 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

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 基本功能
- 操作

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。

适用于 Java 的 SDK 2.x

> **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。

```
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 Reference》中的 listThings。

基本功能

了解基本功能

以下代码示例展示了如何：

- 创建 AWS IoT 事物。
- 生成设备证书。
- 使用属性更新 AWS IoT 事物。
- 返回一个唯一的端点。
- 列出您的 AWS IoT 证书。
- 更新 AWS IoT 阴影。
- 写出状态信息。

- 创建规则。
- 列出您的规则。
- 使用事物名称搜索事物。
- 删除 AWS IoT 事物。

适用于 Java 的 SDK 2.x

> **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。

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 amd delete the certificate.
 * 12. Delete Thing.
 */
public class IotScenario {
```

```
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage =
            """
                Usage:
                    <roleARN> <snsAction>

                Where:
                    roleARN - The ARN of an IAM role that has permission to work
with AWS IOT.
                    snsAction  - An ARN of an SNS topic.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        IotActions iotActions = new IotActions();
        String thingName;
        String ruleName;
        String roleARN = args[0];
        String snsAction = args[1];
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS IoT basics scenario.");
        System.out.println("""
            This example program demonstrates various interactions with the AWS
Internet of Things (IoT) Core service. The program guides you through a series of
steps,
            including creating an IoT Thing, generating a device certificate,
updating the Thing with attributes, and so on.
            It utilizes the AWS SDK for Java V2 and incorporates functionality for
creating and managing IoT Things, certificates, rules,
            shadows, and performing searches. The program aims to showcase AWS IoT
capabilities and provides a comprehensive example for
            developers working with AWS IoT in a Java environment.

            Let's get started...

            """);
        System.out.println(DASHES);
```

```
        System.out.println("1. Create an AWS IoT Thing.");
        System.out.println("""
            An AWS IoT Thing represents a virtual entity in the AWS IoT service that
can be associated with
            a physical device.
            """);
        // Prompt the user for input.
        System.out.print("Enter Thing name: ");
        thingName = scanner.nextLine();
        iotActions.createIoTThing(thingName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Generate a device certificate.");
        System.out.println("""
            A device certificate performs a role in securing the communication
between devices (Things)
            and the AWS IoT platform.
            """);

        System.out.print("Do you want to create a certificate for " +thingName +"?
(y/n)");
        String certAns = scanner.nextLine();
        String certificateArn="" ;
        if (certAns != null && certAns.trim().equalsIgnoreCase("y")) {
            certificateArn = iotActions.createCertificate();
            System.out.println("Attach the certificate to the AWS IoT Thing.");
            iotActions.attachCertificateToThing(thingName, certificateArn);
        } else {
            System.out.println("A device certificate was not created.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Update an AWS IoT Thing with Attributes.");
        System.out.println("""
             IoT Thing attributes, represented as key-value pairs, offer a pivotal
advantage in facilitating efficient data
             management and retrieval within the AWS IoT ecosystem.
            """);
        waitForInputToContinue(scanner);
        iotActions.updateShadowThing(thingName);
        waitForInputToContinue(scanner);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Return a unique endpoint specific to the Amazon Web
Services account.");
        System.out.println("""
            An IoT Endpoint refers to a specific URL or Uniform Resource Locator
that serves as the entry point for communication between IoT devices and the AWS
IoT service.
           """);
        waitForInputToContinue(scanner);
        String endpointUrl = iotActions.describeEndpoint();
        System.out.println("The endpoint is "+endpointUrl);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. List your AWS IoT certificates");
        waitForInputToContinue(scanner);
        if (certificateArn.length() > 0) {
            iotActions.listCertificates();
        } else {
            System.out.println("You did not create a certificates. Skipping this
step.");
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Create an IoT shadow that refers to a digital
representation or virtual twin of a physical IoT device");
        System.out.println("""
            A Thing Shadow refers to a feature that enables you to create a virtual
representation, or "shadow,"
            of a physical device or thing. The Thing Shadow allows you to
synchronize and control the state of a device between
            the cloud and the device itself. and the AWS IoT service. For example,
you can write and retrieve JSON data from a Thing Shadow.
           """);
        waitForInputToContinue(scanner);
        iotActions.updateShadowThing(thingName);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("7. Write out the state information, in JSON format.");
        waitForInputToContinue(scanner);
        iotActions.getPayload(thingName);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Creates a rule");
        System.out.println("""
        Creates a rule that is an administrator-level action.
        Any user who has permission to create rules will be able to access data
processed by the rule.
        """);
        System.out.print("Enter Rule name: ");
        ruleName = scanner.nextLine();
        iotActions.createIoTRule(roleARN, ruleName, snsAction);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. List your rules.");
        waitForInputToContinue(scanner);
        iotActions.listIoTRules();
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Search things using the Thing name.");
        waitForInputToContinue(scanner);
        String queryString = "thingName:"+thingName ;
        iotActions.searchThings(queryString);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        if (certificateArn.length() > 0) {
            System.out.print("Do you want to detach and delete the certificate for "
+thingName +"? (y/n)");
            String delAns = scanner.nextLine();
            if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
                System.out.println("11. You selected to detach amd delete the
certificate.");
                waitForInputToContinue(scanner);
```

```
                iotActions.detachThingPrincipal(thingName, certificateArn);
                iotActions.deleteCertificate(certificateArn);
                waitForInputToContinue(scanner);
            } else {
                System.out.println("11. You selected not to delete the
certificate.");
            }
        } else {
            System.out.println("11. You did not create a certificate so there is
nothing to delete.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Delete the AWS IoT Thing.");
        System.out.print("Do you want to delete the IoT Thing? (y/n)");
        String delAns = scanner.nextLine();
        if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
            iotActions.deleteIoTThing(thingName);
        } else {
            System.out.println("The IoT Thing was not deleted.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The AWS IoT workflow has successfully completed.");
        System.out.println(DASHES);
    }


    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
```

```
        }
    }
}
```

AWS IoT SDK 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)
```

```
                .build();
        }
        return iotAsyncDataPlaneClient;
    }


    private static IotAsyncClient getAsyncClient() {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
            .connectionTimeout(Duration.ofSeconds(60))
            .readTimeout(Duration.ofSeconds(60))
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))
            .apiCallAttemptTimeout(Duration.ofSeconds(90))
            .retryPolicy(RetryPolicy.builder()
                .numRetries(3)
                .build())
            .build();

        if (iotAsyncClient == null) {
            iotAsyncClient = IotAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return iotAsyncClient;
    }

    /**
     * Creates an IoT certificate asynchronously.
     *
     * @return The ARN of the created certificate.
     * <p>
     * This method initiates an asynchronous request to create an IoT certificate.
     * If the request is successful, it prints the certificate details and returns
the certificate ARN.
     * If an exception occurs, it prints the error message.
     */
    public String createCertificate() {
```

```
        CompletableFuture<CreateKeysAndCertificateResponse> future =
getAsyncClient().createKeysAndCertificate();
        final String[] certificateArn = {null};
        future.whenComplete((response, ex) -> {
            if (response != null) {
                String certificatePem = response.certificatePem();
                certificateArn[0] = response.certificateArn();

                // Print the details.
                System.out.println("\nCertificate:");
                System.out.println(certificatePem);
                System.out.println("\nCertificate ARN:");
                System.out.println(certificateArn[0]);

            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + cause.getMessage());
                }
            }
        });

        future.join();
        return certificateArn[0];
    }

    /**
     * Creates an IoT Thing with the specified name asynchronously.
     *
     * @param thingName The name of the IoT Thing to create.
     *
     * This method initiates an asynchronous request to create an IoT Thing with the
specified name.
     * If the request is successful, it prints the name of the thing and its ARN
value.
     * If an exception occurs, it prints the error message.
     */
    public void createIoTThing(String thingName) {
        CreateThingRequest createThingRequest = CreateThingRequest.builder()
            .thingName(thingName)
```

```
            .build();

        CompletableFuture<CreateThingResponse> future =
getAsyncClient().createThing(createThingRequest);
        future.whenComplete((createThingResponse, ex) -> {
            if (createThingResponse != null) {
                System.out.println(thingName + " was successfully created. The ARN
value is " + createThingResponse.thingArn());
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + cause.getMessage());
                }
            }
        });

        future.join();
    }

    /**
     * Attaches a certificate to an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     * @param certificateArn The ARN of the certificate to attach.
     *
     * This method initiates an asynchronous request to attach a certificate to an
IoT Thing.
     * If the request is successful, it prints a confirmation message and additional
information about the Thing.
     * If an exception occurs, it prints the error message.
     */
    public void attachCertificateToThing(String thingName, String certificateArn) {
        AttachThingPrincipalRequest principalRequest =
AttachThingPrincipalRequest.builder()
            .thingName(thingName)
            .principal(certificateArn)
            .build();

        CompletableFuture<AttachThingPrincipalResponse> future =
getAsyncClient().attachThingPrincipal(principalRequest);
        future.whenComplete((attachResponse, ex) -> {
```

```
            if (attachResponse != null &&
attachResponse.sdkHttpResponse().isSuccessful()) {
                System.out.println("Certificate attached to Thing successfully.");

                // Print additional information about the Thing.
                describeThing(thingName);
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to attach certificate to Thing. HTTP
Status Code: " +
                            attachResponse.sdkHttpResponse().statusCode());
                }
            }
        });

        future.join();
    }

    /**
     * Describes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to describe an IoT Thing.
     * If the request is successful, it prints the Thing details.
     * If an exception occurs, it prints the error message.
     */
    private void describeThing(String thingName) {
        DescribeThingRequest thingRequest = DescribeThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DescribeThingResponse> future =
getAsyncClient().describeThing(thingRequest);
        future.whenComplete((describeResponse, ex) -> {
            if (describeResponse != null) {
                System.out.println("Thing Details:");
                System.out.println("Thing Name: " + describeResponse.thingName());
```

```
                System.out.println("Thing ARN: " + describeResponse.thingArn());
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to describe Thing.");
                }
            }
        });

        future.join();
    }

    /**
     * Updates the shadow of an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to update the shadow of an IoT
Thing.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void updateShadowThing(String thingName) {
        // Create Thing Shadow State Document.
        String stateDocument = "{\"state\":{\"reported\":{\"temperature\":25,
\"humidity\":50}}}";
        SdkBytes data = SdkBytes.fromString(stateDocument, StandardCharsets.UTF_8);
        UpdateThingShadowRequest updateThingShadowRequest =
UpdateThingShadowRequest.builder()
            .thingName(thingName)
            .payload(data)
            .build();

        CompletableFuture<UpdateThingShadowResponse> future =
getAsyncDataPlaneClient().updateThingShadow(updateThingShadowRequest);
        future.whenComplete((updateResponse, ex) -> {
            if (updateResponse != null) {
                System.out.println("Thing Shadow updated successfully.");
            } else {
```

```
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to update Thing Shadow.");
                }
            }
        });

        future.join();
    }

    /**
     * Describes the endpoint of the IoT service asynchronously.
     *
     * @return A CompletableFuture containing the full endpoint URL.
     *
     * This method initiates an asynchronous request to describe the endpoint of the
 IoT service.
     * If the request is successful, it prints and returns the full endpoint URL.
     * If an exception occurs, it prints the error message.
     */
    public String describeEndpoint() {
        CompletableFuture<DescribeEndpointResponse> future =
 getAsyncClient().describeEndpoint(DescribeEndpointRequest.builder().endpointType("iot:Data-
ATS").build());
        final String[] result = {null};

        future.whenComplete((endpointResponse, ex) -> {
            if (endpointResponse != null) {
                String endpointUrl = endpointResponse.endpointAddress();
                String exString = getValue(endpointUrl);
                String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";

                System.out.println("Full Endpoint URL: " + fullEndpoint);
                result[0] = fullEndpoint;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
 ex.getCause() : ex;
                if (cause instanceof IotException) {
```

```
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + cause.getMessage());
                }
            }
        });

        future.join();
        return result[0];
    }

    /**
     * Extracts a specific value from the endpoint URL.
     *
     * @param input The endpoint URL to process.
     * @return The extracted value from the endpoint URL.
     */
    private static String getValue(String input) {
        // Define a regular expression pattern for extracting the subdomain.
        Pattern pattern = Pattern.compile("^(.*?)\\.iot\\.us-east-1\\.amazonaws\
\.com");

        // Match the pattern against the input string.
        Matcher matcher = pattern.matcher(input);

        // Check if a match is found.
        if (matcher.find()) {
            // Extract the subdomain from the first capturing group.
            String subdomain = matcher.group(1);
            System.out.println("Extracted subdomain: " + subdomain);
            return subdomain ;
        } else {
            System.out.println("No match found");
        }
        return "" ;
    }

    /**
     * Lists all certificates asynchronously.
     *
     * This method initiates an asynchronous request to list all certificates.
     * If the request is successful, it prints the certificate IDs and ARNs.
     * If an exception occurs, it prints the error message.
```

```
     */
    public void listCertificates() {
        CompletableFuture<ListCertificatesResponse> future =
getAsyncClient().listCertificates();
        future.whenComplete((response, ex) -> {
            if (response != null) {
                List<Certificate> certList = response.certificates();
                for (Certificate cert : certList) {
                    System.out.println("Cert id: " + cert.certificateId());
                    System.out.println("Cert Arn: " + cert.certificateArn());
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to list certificates.");
                }
            }
        });

        future.join();
    }

    /**
     * Retrieves the payload of a Thing's shadow asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     *
     * This method initiates an asynchronous request to get the payload of a Thing's
shadow.
     * If the request is successful, it prints the shadow data.
     * If an exception occurs, it prints the error message.
     */
    public void getPayload(String thingName) {
        GetThingShadowRequest getThingShadowRequest =
GetThingShadowRequest.builder()
            .thingName(thingName)
            .build();
```

```
        CompletableFuture<GetThingShadowResponse> future =
getAsyncDataPlaneClient().getThingShadow(getThingShadowRequest);
        future.whenComplete((getThingShadowResponse, ex) -> {
            if (getThingShadowResponse != null) {
                // Extracting payload from response.
                SdkBytes payload = getThingShadowResponse.payload();
                String payloadString = payload.asUtf8String();
                System.out.println("Received Shadow Data: " + payloadString);
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to get Thing Shadow payload.");
                }
            }
        });

        future.join();
    }

    /**
     * Creates an IoT rule asynchronously.
     *
     * @param roleARN The ARN of the IAM role that grants access to the rule's
actions.
     * @param ruleName The name of the IoT rule.
     * @param action The ARN of the action to perform when the rule is triggered.
     *
     * This method initiates an asynchronous request to create an IoT rule.
     * If the request is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void createIoTRule(String roleARN, String ruleName, String action) {
        String sql = "SELECT * FROM '" + TOPIC + "'";
        SnsAction action1 = SnsAction.builder()
            .targetArn(action)
            .roleArn(roleARN)
            .build();

        // Create the action.
```

```
        Action myAction = Action.builder()
            .sns(action1)
            .build();

        // Create the topic rule payload.
        TopicRulePayload topicRulePayload = TopicRulePayload.builder()
            .sql(sql)
            .actions(myAction)
            .build();

        // Create the topic rule request.
        CreateTopicRuleRequest topicRuleRequest = CreateTopicRuleRequest.builder()
            .ruleName(ruleName)
            .topicRulePayload(topicRulePayload)
            .build();

        CompletableFuture<CreateTopicRuleResponse> future =
getAsyncClient().createTopicRule(topicRuleRequest);
        future.whenComplete((response, ex) -> {
            if (response != null) {
                System.out.println("IoT Rule created successfully.");
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to create IoT Rule.");
                }
            }
        });

        future.join();
    }

    /**
     * Lists IoT rules asynchronously.
     *
     * This method initiates an asynchronous request to list IoT rules.
     * If the request is successful, it prints the names and ARNs of the rules.
     * If an exception occurs, it prints the error message.
     */
```

```
    public void listIoTRules() {
        ListTopicRulesRequest listTopicRulesRequest =
ListTopicRulesRequest.builder().build();
        CompletableFuture<ListTopicRulesResponse> future =
getAsyncClient().listTopicRules(listTopicRulesRequest);
        future.whenComplete((listTopicRulesResponse, ex) -> {
            if (listTopicRulesResponse != null) {
                System.out.println("List of IoT Rules:");
                List<TopicRuleListItem> ruleList = listTopicRulesResponse.rules();
                for (TopicRuleListItem rule : ruleList) {
                    System.out.println("Rule Name: " + rule.ruleName());
                    System.out.println("Rule ARN: " + rule.ruleArn());
                    System.out.println("--------------");
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to list IoT Rules.");
                }
            }
        });

        future.join();
    }

    /**
     * Searches for IoT Things asynchronously based on a query string.
     *
     * @param queryString The query string to search for Things.
     *
     * This method initiates an asynchronous request to search for IoT Things.
     * If the request is successful and Things are found, it prints their IDs.
     * If no Things are found, it prints a message indicating so.
     * If an exception occurs, it prints the error message.
     */
    public void searchThings(String queryString) {
        SearchIndexRequest searchIndexRequest = SearchIndexRequest.builder()
            .queryString(queryString)
            .build();
```

```
        CompletableFuture<SearchIndexResponse> future =
getAsyncClient().searchIndex(searchIndexRequest);
        future.whenComplete((searchIndexResponse, ex) -> {
            if (searchIndexResponse != null) {
                // Process the result.
                if (searchIndexResponse.things().isEmpty()) {
                    System.out.println("No things found.");
                } else {
                    searchIndexResponse.things().forEach(thing ->
System.out.println("Thing id found using search is " + thing.thingId()));
                }
            } else {
                Throwable cause = ex != null ? ex.getCause() : null;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else if (cause != null) {
                    System.err.println("Unexpected error: " + cause.getMessage());
                } else {
                    System.err.println("Failed to search for IoT Things.");
                }
            }
        });

        future.join();
    }

    /**
     * Detaches a principal (certificate) from an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing.
     * @param certificateArn The ARN of the certificate to detach.
     *
     * This method initiates an asynchronous request to detach a certificate from an
IoT Thing.
     * If the detachment is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void detachThingPrincipal(String thingName, String certificateArn) {
        DetachThingPrincipalRequest thingPrincipalRequest =
DetachThingPrincipalRequest.builder()
            .principal(certificateArn)
            .thingName(thingName)
```

```
            .build();

        CompletableFuture<DetachThingPrincipalResponse> future =
getAsyncClient().detachThingPrincipal(thingPrincipalRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println(certificateArn + " was successfully removed from
" + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }

    /**
     * Deletes a certificate asynchronously.
     *
     * @param certificateArn The ARN of the certificate to delete.
     *
     * This method initiates an asynchronous request to delete a certificate.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteCertificate(String certificateArn) {
        DeleteCertificateRequest certificateProviderRequest =
DeleteCertificateRequest.builder()
            .certificateId(extractCertificateId(certificateArn))
            .build();

        CompletableFuture<DeleteCertificateResponse> future =
getAsyncClient().deleteCertificate(certificateProviderRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println(certificateArn + " was successfully deleted.");
            } else {
                Throwable cause = ex.getCause();
```

```
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }

    /**
     * Deletes an IoT Thing asynchronously.
     *
     * @param thingName The name of the IoT Thing to delete.
     *
     * This method initiates an asynchronous request to delete an IoT Thing.
     * If the deletion is successful, it prints a confirmation message.
     * If an exception occurs, it prints the error message.
     */
    public void deleteIoTThing(String thingName) {
        DeleteThingRequest deleteThingRequest = DeleteThingRequest.builder()
            .thingName(thingName)
            .build();

        CompletableFuture<DeleteThingResponse> future =
getAsyncClient().deleteThing(deleteThingRequest);
        future.whenComplete((voidResult, ex) -> {
            if (ex == null) {
                System.out.println("Deleted Thing " + thingName);
            } else {
                Throwable cause = ex.getCause();
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + ex.getMessage());
                }
            }
        });

        future.join();
    }
```

```
    // Get the cert Id  from the Cert ARN value.
    private String extractCertificateId(String certificateArn) {
        // Example ARN: arn:aws:iot:region:account-id:cert/certificate-id.
        String[] arnParts = certificateArn.split(":");
        String certificateIdPart = arnParts[arnParts.length - 1];
        return certificateIdPart.substring(certificateIdPart.lastIndexOf("/") + 1);
    }
}
```

操作

`AttachThingPrincipal`

以下代码示例演示了如何使用 `AttachThingPrincipal`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[AttachThingPrincipal](#)中的。

CreateKeysAndCertificate

以下代码示例演示了如何使用 CreateKeysAndCertificate。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateKeysAndCertificate中的。

`CreateThing`

以下代码示例演示了如何使用 `CreateThing`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateThing中的。

`CreateTopicRule`

以下代码示例演示了如何使用 `CreateTopicRule`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateTopicRule中的。

`DeleteCertificate`

以下代码示例演示了如何使用 `DeleteCertificate`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteCertificate中的。

`DeleteThing`

以下代码示例演示了如何使用 `DeleteThing`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteThing中的。

`DescribeEndpoint`

以下代码示例演示了如何使用 `DescribeEndpoint`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Describes the endpoint of the IoT service asynchronously.
     *
     * @return A CompletableFuture containing the full endpoint URL.
     *
     * This method initiates an asynchronous request to describe the endpoint of the
IoT service.
     * If the request is successful, it prints and returns the full endpoint URL.
     * If an exception occurs, it prints the error message.
     */
    public String describeEndpoint() {
```

```
        CompletableFuture<DescribeEndpointResponse> future =
 getAsyncClient().describeEndpoint(DescribeEndpointRequest.builder().endpointType("iot:Data-
ATS").build());
        final String[] result = {null};

        future.whenComplete((endpointResponse, ex) -> {
            if (endpointResponse != null) {
                String endpointUrl = endpointResponse.endpointAddress();
                String exString = getValue(endpointUrl);
                String fullEndpoint = "https://" + exString + "-ats.iot.us-
east-1.amazonaws.com";

                System.out.println("Full Endpoint URL: " + fullEndpoint);
                result[0] = fullEndpoint;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
 ex.getCause() : ex;
                if (cause instanceof IotException) {
                    System.err.println(((IotException)
 cause).awsErrorDetails().errorMessage());
                } else {
                    System.err.println("Unexpected error: " + cause.getMessage());
                }
            }
        });

        future.join();
        return result[0];
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeEndpoint中的。

DescribeThing

以下代码示例演示了如何使用 DescribeThing。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeThing中的。

`DetachThingPrincipal`

以下代码示例演示了如何使用 `DetachThingPrincipal`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DetachThingPrincipal中的。

`ListCertificates`

以下代码示例演示了如何使用 `ListCertificates`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ListCertificates](#)中的。

`SearchIndex`

以下代码示例演示了如何使用 `SearchIndex`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SearchIndex中的。

AWS IoT data 使用适用于 Java 的 SDK 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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从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`

以下代码示例演示了如何使用 `GetThingShadow`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetThingShadow中的。

`UpdateThingShadow`

以下代码示例演示了如何使用 `UpdateThingShadow`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[UpdateThingShadow](#)中的。

AWS IoT FleetWise 使用适用于 Java 的 SDK 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 FleetWise。 AWS SDK for Java 2.x

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 基本功能
- 操作

开始使用

你好 AWS IoT FleetWise

以下代码示例展示了如何开始使用 AWS IoT FleetWise。

适用于 Java 的 SDK 2.x

> **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。

```
public class HelloFleetwise {

        public static void main(String[] args) {
            ListSignalCatalogs();
        }
```

```
        public static void  ListSignalCatalogs() {
            try (IoTFleetWiseClient fleetWiseClient = IoTFleetWiseClient.builder()
                    .region(Region.US_EAST_1)
                    .credentialsProvider(DefaultCredentialsProvider.create())
                    .build()) {

                ListSignalCatalogsRequest request =
ListSignalCatalogsRequest.builder()
                        .maxResults(10) // Optional: limit per page
                        .build();

                ListSignalCatalogsIterable paginator =
fleetWiseClient.listSignalCatalogsPaginator(request);
                boolean found = false;

                for (ListSignalCatalogsResponse response : paginator) {
                    for (SignalCatalogSummary summary : response.summaries()) {
                        found = true;
                        System.out.println("Catalog Name: " + summary.name());
                        System.out.println("ARN: " + summary.arn());
                        System.out.println("Created: " + summary.creationTime());
                        System.out.println("Last Modified: " +
summary.lastModificationTime());
                        System.out.println("---------------");
                    }
                }

                if (!found) {
                    System.out.println("No AWS Fleetwise Signal Catalogs were
found.");
                }

            } catch (IoTFleetWiseException e) {
                System.err.println("Error listing signal catalogs: " +
e.awsErrorDetails().errorMessage());
                throw new RuntimeException(e);
            }
        }
    }
```

- 有关 API 的详细信息，请参阅 AP AWS SDK for Java 2.x I 参考[listSignalCatalogs中的分页器](#)。

基本功能

了解基本功能

以下代码示例展示了如何：

- 创建标准化信号的集合。
- 创建表示一组车辆的实例集。
- 创建模型清单。
- 创建解码器清单。
- 检查模型清单的状态。
- 检查解码器的状态。
- 创建 IoT 事物。
- 创建车辆。
- 显示车辆详细信息。
- 删除资 AWS IoT FleetWise 产。

适用于 Java 的 SDK 2.x

> **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。

运行演示 AWS IoT SiteWise 功能的交互式场景。

```
public class FleetwiseScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    static FleetwiseActions actions = new FleetwiseActions();
    private static final Logger logger =
 LoggerFactory.getLogger(FleetwiseScenario.class);
    static Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        final String usage =
                """
                Usage:
```

```
                    <signalCatalogName> <manifestName> <fleetId> <vecName> <decName>

                Where:
                    signalCatalogName     - The name of the Signal Catalog to create
(eg, catalog30).
                    manifestName          - The name of the Vehicle Model (Model
Manifest) to create (eg, manifest30).
                    fleetId               - The ID of the Fleet to create (eg,
fleet30).
                    vecName               - The name of the Vehicle to create (eg,
vehicle30).
                    decName               - The name of the Decoder Manifest to
create (eg, decManifest30).
                """;

        if (args.length != 5) {
            logger.info(usage);
            return;
        }

        String signalCatalogName = args[0];
        String manifestName = args[1];
        String fleetId = args[2];
        String vecName = args[3];
        String decName = args[4];

        logger.info(
                """
                 AWS IoT FleetWise is a managed service that simplifies the
                 process of collecting, organizing, and transmitting vehicle
                 data to the cloud in near real-time. Designed for automakers
                 and fleet operators, it allows you to define vehicle models,
                 specify the exact data you want to collect (such as engine
                 temperature, speed, or battery status), and send this data to
                 AWS for analysis. By using intelligent data collection
                 techniques, IoT FleetWise reduces the volume of data
                 transmitted by filtering and transforming it at the edge,
                 helping to minimize bandwidth usage and costs.

                At its core, AWS IoT FleetWise helps organizations build
                scalable systems for vehicle data management and analytics,
                supporting a wide variety of vehicles and sensor configurations.
                You can define signal catalogs and decoder manifests that describe
                how raw CAN bus signals are translated into readable data, making
```

```
                the platform highly flexible and extensible. This allows
                manufacturers to optimize vehicle performance, improve safety,
                and reduce maintenance costs by gaining real-time visibility
                into fleet operations.
                """);

        waitForInputToContinue(scanner);
        logger.info(DASHES);
        try {
            runScenario(signalCatalogName, manifestName, fleetId, vecName, decName);
        } catch (RuntimeException e) {
            logger.info(e.getMessage());
        }
    }

    private static void runScenario(String signalCatalogName,
                                    String manifestName,
                                    String fleetId,
                                    String vecName,
                                    String decName) {
        logger.info(DASHES);
        logger.info("1. Creates a collection of standardized signals that can be
reused to create vehicle models");
        String signalCatalogArn;
        try {
            signalCatalogArn =
actions.createSignalCatalogAsync(signalCatalogName).join();
            logger.info("The collection ARN is " + signalCatalogArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ValidationException) {
                logger.error("The request failed due to a validation issue: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Create a fleet that represents a group of vehicles");
        logger.info(
```

```
                """
                Creating an IoT FleetWise fleet allows you to efficiently collect,
                organize, and transfer vehicle data to the cloud, enabling real-
time
                insights into vehicle performance and health.

                It helps reduce data costs by allowing you to filter and prioritize
                only the most relevant vehicle signals, supporting advanced
 analytics
                and predictive maintenance use cases.
                """);

        waitForInputToContinue(scanner);
        String fleetid;
        try {
            fleetid = actions.createFleetAsync(signalCatalogArn, fleetId).join();
            logger.info("The fleet Id is " + fleetid);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The resource was not found: {}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Create a model manifest");
        logger.info(
                """
                An AWS IoT FleetWise manifest defines the structure and
                relationships of vehicle data. The model manifest specifies
                which signals to collect and how they relate to vehicle systems,
                while the decoder manifest defines how to decode raw vehicle data
                into meaningful signals.
                """);
        waitForInputToContinue(scanner);
        String manifestArn;
        try {
            List<Node> nodes =
 actions.listSignalCatalogNodeAsync(signalCatalogName).join();
```

```
            manifestArn = actions.createModelManifestAsync(manifestName,
signalCatalogArn, nodes).join();
            logger.info("The manifest ARN is {}", manifestArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            logger.error("An unexpected error occurred.", cause);
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Create a decoder manifest");
        logger.info(
                """
                A decoder manifest in AWS IoT FleetWise defines how raw vehicle
                data (such as CAN signals) should be interpreted and decoded
                into meaningful signals. It acts as a translation layer
                that maps vehicle-specific protocols to standardized data formats
                using decoding rules. This is crucial for extracting usable
                data from different vehicle models, even when their data
                formats vary.

                """);
        waitForInputToContinue(scanner);
        String decArn;
        try {
            decArn = actions.createDecoderManifestAsync(decName,
manifestArn).join();
            logger.info("The decoder manifest ARN is {}", decArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            logger.error("An unexpected error occurred.", cause);
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("5. Check the status of the model manifest");
        logger.info(
                """
                The model manifest must be in an ACTIVE state before it can be used
                to create or update a vehicle.
                """);
```

```
        waitForInputToContinue(scanner);
        try {
            actions.updateModelManifestAsync(manifestName);
            actions.waitForModelManifestActiveAsync(manifestName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            logger.error("An unexpected error occurred while waiting for the model
manifest status.", cause);
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("6. Check the status of the decoder");
        logger.info(
                """
                The decoder manifest must be in an ACTIVE state before it can be
used
                to create or update a vehicle.
                """);
        waitForInputToContinue(scanner);
        try {
            actions.updateDecoderManifestAsync(decName);
            actions.waitForDecoderManifestActiveAsync(decName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            logger.error("An unexpected error occurred while waiting for the decoder
manifest status.", cause);
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Create an IoT Thing");
        logger.info(
                """
                AWS IoT FleetWise expects an existing AWS IoT Thing with the same
                name as the vehicle name you are passing to createVehicle method.
                Before calling createVehicle(), you must create an AWS IoT Thing
                with the same name using the AWS IoT Core service.
                """);
        waitForInputToContinue(scanner);
        try {
```

```
            actions.createThingIfNotExistsAsync(vecName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceAlreadyExistsException) {
                logger.error("The resource exists: {}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Create a vehicle");
        logger.info(
                """
                Creating a vehicle in AWS IoT FleetWise allows you to digitally
                represent and manage a physical vehicle within the AWS ecosystem.
                This enables efficient ingestion, transformation, and transmission
                of vehicle telemetry data to the cloud for analysis.
                """);
        waitForInputToContinue(scanner);
        try {
            actions.createVehicleAsync(vecName, manifestArn, decArn).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();

            if (cause instanceof ResourceNotFoundException) {
                logger.error("The required resource was not found: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred while creating vehicle.",
cause);
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("9. Display vehicle details");
        waitForInputToContinue(scanner);
        try {
            actions.getVehicleDetailsAsync(vecName).join();
```

```
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The resource was not found: {}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("10. Delete the AWS IoT Fleetwise Assets");
        logger.info("Would you like to delete the IoT Fleetwise Assets? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            try {
                actions.deleteVehicleAsync(vecName).join();
                actions.deleteDecoderManifestAsync(decName).join();
                actions.deleteModelManifestAsync(manifestName).join();
                actions.deleteFleetAsync(fleetid).join();
                actions.deleteSignalCatalogAsync(signalCatalogName).join();
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    // Handle the case where the resource is not found.
                    logger.error("The resource was not found: {}",
cause.getMessage());
                } else if (cause instanceof RuntimeException) {
                    // Handle other runtime exceptions.
                    logger.error("An unexpected error occurred: {}",
cause.getMessage());
                } else {
                    // Catch any other unexpected exceptions.
                    logger.error("An unknown error occurred.", cause);
                }
                return;
            }

            logger.info(DASHES);
            logger.info(
                    """
```

```
                    Thank you for checking out the AWS IoT Fleetwise Service Use
 demo. We hope you
                    learned something new, or got some inspiration for your own apps
 today.
                    For more AWS code examples, have a look at:
                    https://docs.aws.amazon.com/code-library/latest/ug/what-is-code-
library.html
                    """);
            logger.info(DASHES);
        } else {
            logger.info("The AWS resources will not be deleted.");
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

AWS IoT FleetWise SDK 方法的包装器类。

```
public class FleetwiseActions {
    private static final Logger logger =
 LoggerFactory.getLogger(FleetwiseActions.class);
    private static IoTFleetWiseAsyncClient ioTFleetWiseAsyncClient;

    private static IoTFleetWiseAsyncClient getAsyncClient() {
        if (ioTFleetWiseAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                    .maxConcurrency(100)
```

```
                    .connectionTimeout(Duration.ofSeconds(60))
                    .readTimeout(Duration.ofSeconds(60))
                    .writeTimeout(Duration.ofSeconds(60))
                    .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                    .apiCallTimeout(Duration.ofMinutes(2))
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))
                    .retryStrategy(RetryMode.STANDARD)
                    .build();

            ioTFleetWiseAsyncClient = IoTFleetWiseAsyncClient.builder()
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
        return ioTFleetWiseAsyncClient;
    }


    /**
     * Creates a signal catalog.
     *
     * @param signalCatalogName the name of the signal catalog to be created
     * @return a {@link CompletableFuture} that completes with the Amazon Resource
Name (ARN) of the created signal catalog
     */
    public CompletableFuture<String> createSignalCatalogAsync(String
signalCatalogName) {
        return deleteSignalCatalogIfExistsAsync(signalCatalogName)
                .thenCompose(ignored -> delayAsync(2000)) // Wait for 2 seconds
                .thenCompose(ignored -> {
                    List<Node> nodes = List.of(
                            Node.builder().branch(
                                    Branch.builder()
                                            .fullyQualifiedName("Vehicle")
                                            .description("Root branch")
                                            .build()
                            ).build(),
                            Node.builder().branch(
                                    Branch.builder()

.fullyQualifiedName("Vehicle.Powertrain")
```

```
                                        .description("Powertrain branch")
                                        .build()
                            ).build(),
                            Node.builder().sensor(
                                    Sensor.builder()

.fullyQualifiedName("Vehicle.Powertrain.EngineRPM")
                                        .description("Engine RPM")
                                        .dataType(NodeDataType.DOUBLE)
                                        .unit("rpm")
                                        .build()
                            ).build(),
                            Node.builder().sensor(
                                    Sensor.builder()

.fullyQualifiedName("Vehicle.Powertrain.VehicleSpeed")
                                        .description("Vehicle Speed")
                                        .dataType(NodeDataType.DOUBLE)
                                        .unit("km/h")
                                        .build()
                            ).build()
                    );

                    CreateSignalCatalogRequest request =
CreateSignalCatalogRequest.builder()
                            .name(signalCatalogName)
                            .nodes(nodes)
                            .build();

                    CompletableFuture<String> result = new CompletableFuture<>();

                    getAsyncClient().createSignalCatalog(request)
                            .whenComplete((response, exception) -> {
                                if (exception != null) {
                                    Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                                    if (cause instanceof ValidationException) {
                                        result.completeExceptionally(cause);
                                    } else {
                                        result.completeExceptionally(new
RuntimeException("Error creating the catalog", cause));
                                    }
                                } else {
```

```
                                    result.complete(response.arn());
                                }
                            });

                    return result;
                });
    }

    /**
     * Delays the execution of the current thread asynchronously for the specified
duration.
     *
     * @param millis the duration of the delay in milliseconds
     * @return a {@link CompletableFuture} that completes after the specified delay
     */
    private static CompletableFuture<Void> delayAsync(long millis) {
        return CompletableFuture.runAsync(() -> {
            try {
                Thread.sleep(millis);
            } catch (InterruptedException e) {
                throw new CompletionException("Sleep interrupted", e);
            }
        });
    }

    /**
     * Deletes the specified signal catalog.
     *
     * @param signalCatalogName the name of the signal catalog to delete
     * @return a {@link CompletableFuture} representing the asynchronous operation.
     */
    public static CompletableFuture<Void> deleteSignalCatalogIfExistsAsync(String
signalCatalogName) {
        DeleteSignalCatalogRequest request = DeleteSignalCatalogRequest.builder()
                .name(signalCatalogName)
                .build();

        return getAsyncClient().deleteSignalCatalog(request)
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ResourceNotFoundException) {
```

```
                            throw new CompletionException(new
RuntimeException("Signal Catalog not found: " + signalCatalogName));
                        }
                        throw new RuntimeException("Failed to delete signal catalog:
" + signalCatalogName, cause);
                    }
                    return null;
                });
    }


    /**
     * Creates a new decoder manifest.
     *
     * @param name             the name of the decoder manifest
     * @param modelManifestArn the ARN of the model manifest
     * @return a {@link CompletableFuture} that completes with the ARN of the
created decoder manifest
     */
    public CompletableFuture<String> createDecoderManifestAsync(String name, String
modelManifestArn) {
        String interfaceId = "can0";
        NetworkInterface networkInterface = NetworkInterface.builder()
                .interfaceId(interfaceId)
                .type(NetworkInterfaceType.CAN_INTERFACE)
                .canInterface(CanInterface.builder()
                        .name("canInterface0")
                        .protocolName("CAN")
                        .protocolVersion("1.0")
                        .build())
                .build();

        // Vehicle.Powertrain.EngineRPM decoder.
        SignalDecoder engineRpmDecoder = SignalDecoder.builder()
                .fullyQualifiedName("Vehicle.Powertrain.EngineRPM")
                .interfaceId(interfaceId)
                .type(SignalDecoderType.CAN_SIGNAL)
                .canSignal(CanSignal.builder()
                        .messageId(100)
                        .isBigEndian(false)
                        .isSigned(false)
                        .startBit(0)
                        .length(16)
                        .factor(1.0)
```

```
                        .offset(0.0)
                        .build())
                .build();

        // Vehicle.Powertrain.VehicleSpeed decoder.
        SignalDecoder vehicleSpeedDecoder = SignalDecoder.builder()
                .fullyQualifiedName("Vehicle.Powertrain.VehicleSpeed")
                .interfaceId(interfaceId)
                .type(SignalDecoderType.CAN_SIGNAL)
                .canSignal(CanSignal.builder()
                        .messageId(101)
                        .isBigEndian(false)
                        .isSigned(false)
                        .startBit(16)
                        .length(16)
                        .factor(1.0)
                        .offset(0.0)
                        .build())
                .build();

        CreateDecoderManifestRequest request =
CreateDecoderManifestRequest.builder()
                .name(name)
                .modelManifestArn(modelManifestArn)
                .networkInterfaces(List.of(networkInterface))
                .signalDecoders(List.of(engineRpmDecoder, vehicleSpeedDecoder))
                .build();

        CompletableFuture<String> result = new CompletableFuture<>();

        getAsyncClient().createDecoderManifest(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof DecoderManifestValidationException) {
                            result.completeExceptionally(new
CompletionException("The request contains signal decoders with validation errors: "
+ cause.getMessage(), cause));
                        } else {
                            result.completeExceptionally(new
CompletionException("Failed to create decoder manifest: " + exception.getMessage(),
exception));
```

```
                        }
                    } else {
                        result.complete(response.arn()); // Complete successfully
with the ARN
                    }
                });

        return result;
    }

    /**
     * Deletes a decoder manifest.
     *
     * @param name the name of the decoder manifest to delete
     * @return a {@link CompletableFuture} that completes when the decoder manifest
has been deleted
     */
    public CompletableFuture<Void> deleteDecoderManifestAsync(String name) {
        return
getAsyncClient().deleteDecoderManifest(DeleteDecoderManifestRequest.builder().name(name).bu
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }
                        throw new RuntimeException("Failed to delete the decoder
manifest: " + cause);
                    }
                    return null;
                });
    }


    /**
     * Deletes a vehicle with the specified name.
     *
     * @param vecName the name of the vehicle to be deleted
     * @return a {@link CompletableFuture} that completes when the vehicle has been
deleted
     */
    public CompletableFuture<Void> deleteVehicleAsync(String vecName) {
        DeleteVehicleRequest request = DeleteVehicleRequest.builder()
```

```
                .vehicleName(vecName)
                .build();

        return getAsyncClient().deleteVehicle(request)
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }
                        throw new RuntimeException("Failed to delete the vehicle: "
+ cause);
                    }
                    return null;
                });
    }


    /**
     * Updates the model manifest.
     *
     * @param name the name of the model manifest to update
     */
    public void updateModelManifestAsync(String name) {
        UpdateModelManifestRequest request = UpdateModelManifestRequest.builder()
                .name(name)
                .status(ManifestStatus.ACTIVE)
                .build();

        getAsyncClient().updateModelManifest(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        throw new CompletionException("Failed to update model
manifest: " + exception.getMessage(), exception);
                    }
                })
                .thenApply(response -> null);
    }


    /**
     * Updates the decoder manifest with the given name.
     *
```

```
    * @param name the name of the decoder manifest to update
    * @return a {@link CompletableFuture} that completes when the update operation
is finished
    */
   public CompletableFuture<Void> updateDecoderManifestAsync(String name) {
       UpdateDecoderManifestRequest request =
UpdateDecoderManifestRequest.builder()
               .name(name)
               .status(ManifestStatus.ACTIVE)
               .build();

       return getAsyncClient().updateDecoderManifest(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       throw new CompletionException("Failed to update decoder
manifest: " + exception.getMessage(), exception);
                   }
               })
               .thenApply(response -> null);
   }


   /**
    * Creates a new vehicle in the system.
    *
    * @param vecName     the name of the vehicle to be created
    * @param manifestArn the Amazon Resource Name (ARN) of the model manifest for
the vehicle
    * @param decArn      the Amazon Resource Name (ARN) of the decoder manifest for
the vehicle
    * @return a {@link CompletableFuture} that completes when the vehicle has been
created, or throws a
    */
   public CompletableFuture<Void> createVehicleAsync(String vecName, String
manifestArn, String decArn) {
       CreateVehicleRequest request = CreateVehicleRequest.builder()
               .vehicleName(vecName)
               .modelManifestArn(manifestArn)
               .decoderManifestArn(decArn)
               .build();

       CompletableFuture<Void> result = new CompletableFuture<>();
       getAsyncClient().createVehicle(request)
               .whenComplete((response, exception) -> {
```

```
                    if (exception != null) {
                        Throwable cause = exception instanceof CompletionException ?
exception.getCause() : exception;

                        if (cause instanceof ResourceNotFoundException) {
                            result.completeExceptionally(cause);
                        } else {
                            result.completeExceptionally(new
RuntimeException("Failed to create vehicle: " + cause.getMessage(), cause));
                        }
                    } else {
                        logger.info("Vehicle '{}' created successfully.", vecName);
                        result.complete(null); // mark future as complete
                    }
                });

        return result;
    }


    /**
     * Waits for the decoder manifest to become active.
     *
     * @param decoderName the name of the decoder to wait for
     * @return a {@link CompletableFuture} that completes when the decoder manifest
becomes active, or exceptionally if an error occurs or the manifest becomes invalid
     */
    public CompletableFuture<Void> waitForDecoderManifestActiveAsync(String
decoderName) {
        CompletableFuture<Void> result = new CompletableFuture<>();

        ScheduledExecutorService scheduler =
Executors.newSingleThreadScheduledExecutor();
        AtomicInteger secondsElapsed = new AtomicInteger(0);
        AtomicReference<ManifestStatus> lastStatus = new
AtomicReference<>(ManifestStatus.DRAFT);

        logger.info(" Elapsed: 0s | Decoder Status: DRAFT");

        final Runnable pollTask = new Runnable() {
            @Override
            public void run() {
                int elapsed = secondsElapsed.incrementAndGet();
```

```
                // Check status every 5 seconds
                if (elapsed % 5 == 0) {
                    GetDecoderManifestRequest request =
GetDecoderManifestRequest.builder()
                            .name(decoderName)
                            .build();

                    getAsyncClient().getDecoderManifest(request)
                            .whenComplete((response, exception) -> {
                                if (exception != null) {
                                    Throwable cause = exception instanceof
CompletionException ? exception.getCause() : exception;

                                    scheduler.shutdown();
                                    if (cause instanceof ResourceNotFoundException)
{
                                        result.completeExceptionally(new
RuntimeException("Decoder manifest not found: " + cause.getMessage(), cause));
                                    } else {
                                        result.completeExceptionally(new
RuntimeException("Error while polling decoder manifest status: " +
exception.getMessage(), exception));
                                    }
                                    return;
                                }

                                ManifestStatus status = response.status();
                                lastStatus.set(status);

                                if (status == ManifestStatus.ACTIVE) {
                                    logger.info("\r Elapsed: {}s | Decoder Status:
ACTIVE", elapsed);
                                    scheduler.shutdown();
                                    result.complete(null);
                                } else if (status == ManifestStatus.INVALID) {
                                    logger.info("\r Elapsed: {}s | Decoder Status:
INVALID", elapsed);
                                    scheduler.shutdown();
                                    result.completeExceptionally(new
RuntimeException("Decoder manifest became INVALID. Cannot proceed."));
                                } else {
                                    logger.info("\r# Elapsed: {}s | Decoder Status:
{}", elapsed, status);
                                }
```

```
                            });
                } else {
                    logger.info("\r Elapsed: {}s | Decoder Status: {}", elapsed,
lastStatus.get());
                }
            }
        };

        // Start the task with an initial delay of 1 second, and repeat every second
        scheduler.scheduleAtFixedRate(pollTask, 1, 1, TimeUnit.SECONDS);
        return result;
    }



    /**
     * Waits for the specified model manifest to become active.
     *
     * @param manifestName the name of the model manifest to wait for
     */
    public CompletableFuture<Void> waitForModelManifestActiveAsync(String
manifestName) {
        CompletableFuture<Void> result = new CompletableFuture<>();

        ScheduledExecutorService scheduler =
Executors.newSingleThreadScheduledExecutor();
        AtomicInteger secondsElapsed = new AtomicInteger(0);
        AtomicReference<ManifestStatus> lastStatus = new
AtomicReference<>(ManifestStatus.DRAFT);

        logger.info("Elapsed: 0s | Status: DRAFT");

        final Runnable pollTask = new Runnable() {
            @Override
            public void run() {
                int elapsed = secondsElapsed.incrementAndGet();

                // Only check status every 5 seconds
                if (elapsed % 5 == 0) {
                    GetModelManifestRequest request =
GetModelManifestRequest.builder()
                            .name(manifestName)
                            .build();
```

```
                    getAsyncClient().getModelManifest(request)
                            .whenComplete((response, exception) -> {
                                if (exception != null) {
                                    Throwable cause = exception instanceof
CompletionException ? exception.getCause() : exception;

                                    scheduler.shutdown();
                                    if (cause instanceof ResourceNotFoundException)
{
                                        result.completeExceptionally(new
RuntimeException("Model manifest not found: " + cause.getMessage(), cause));
                                    } else {
                                        result.completeExceptionally(new
RuntimeException("Error while polling model manifest status: " +
exception.getMessage(), exception));
                                    }
                                    return;
                                }

                                ManifestStatus status = response.status();
                                lastStatus.set(status);

                                if (status == ManifestStatus.ACTIVE) {
                                    logger.info("\rElapsed: {}s | Status: ACTIVE",
elapsed);
                                    scheduler.shutdown();
                                    result.complete(null);
                                } else if (status == ManifestStatus.INVALID) {
                                    logger.info("\rElapsed: {}s | Status: INVALID",
elapsed);
                                    scheduler.shutdown();
                                    result.completeExceptionally(new
RuntimeException("Model manifest became INVALID. Cannot proceed."));
                                } else {
                                    logger.info("\rElapsed: {}s | Status: {}",
elapsed, status);
                                }
                            });
                } else {
                    logger.info("\rElapsed: {}s | Status: {}", elapsed,
lastStatus.get());
                }
            }
        };
```

```
        // Start the task with an initial delay of 1 second, and repeat every second
        scheduler.scheduleAtFixedRate(pollTask, 1, 1, TimeUnit.SECONDS);
        return result;
    }



    /**
     * Fetches the details of a vehicle.
     *
     * @param vehicleName the name of the vehicle to fetch details for
     * @return a {@link CompletableFuture} that completes when the vehicle details
have been fetched
     */
    public CompletableFuture<Void> getVehicleDetailsAsync(String vehicleName) {
        GetVehicleRequest request = GetVehicleRequest.builder()
                .vehicleName(vehicleName)
                .build();

        CompletableFuture<Void> result = new CompletableFuture<>();

        getAsyncClient().getVehicle(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception instanceof CompletionException ?
exception.getCause() : exception;

                        if (cause instanceof ResourceNotFoundException) {
                            result.completeExceptionally(cause); // don't rewrap
                        } else {
                            result.completeExceptionally(new
RuntimeException("Failed to fetch vehicle details: " + cause.getMessage(), cause));
                        }
                    } else {
                        Map<String, Object> details = new HashMap<>();
                        details.put("vehicleName", response.vehicleName());
                        details.put("arn", response.arn());
                        details.put("modelManifestArn",
response.modelManifestArn());
                        details.put("decoderManifestArn",
response.decoderManifestArn());
                        details.put("attributes", response.attributes());
```

```
                        details.put("creationTime",
response.creationTime().toString());
                        details.put("lastModificationTime",
response.lastModificationTime().toString());

                        logger.info("Vehicle Details:");
                        details.forEach((key, value) -> logger.info("• {} : {}",
key, value));

                        result.complete(null); // mark as successful
                    }
                });

        return result;
    }


    /**
     * Creates an IoT Thing if it does not already exist.
     *
     * @param thingName the name of the IoT Thing to create
     * @return a {@link CompletableFuture} that completes when the IoT Thing has
been created or if it already exists
     */
    public CompletableFuture<Void> createThingIfNotExistsAsync(String thingName) {
        IotAsyncClient iotClient = IotAsyncClient.builder()
                .region(Region.US_EAST_1)
                .build();

        CreateThingRequest request = CreateThingRequest.builder()
                .thingName(thingName)
                .build();

        return iotClient.createThing(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        if (exception instanceof ResourceAlreadyExistsException) {
                            logger.info(" IoT Thing already exists: " + thingName);
                        } else {
                            throw new CompletionException("Failed to create IoT
Thing: " + thingName, exception);
                        }
                    } else {
                        logger.info("IoT Thing created: " + response.thingName());
```

```
                    }
                })
                .thenApply(response -> null);
    }


    /**
     * Deletes a model manifest.
     *
     * @param name the name of the model manifest to delete
     * @return a {@link CompletableFuture} that completes when the model manifest
has been deleted
     */
    public CompletableFuture<Void> deleteModelManifestAsync(String name) {
        DeleteModelManifestRequest request = DeleteModelManifestRequest.builder()
                .name(name)
                .build();

        return getAsyncClient().deleteModelManifest(request)
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }
                        throw new RuntimeException("Failed to delete the model
manifest: " + cause);
                    }
                    logger.info("{} was successfully deleted", name);
                    return null;
                });
    }


    /**
     * Deletes a signal catalog.
     *
     * @param name the name of the signal catalog to delete
     * @return a {@link CompletableFuture} that completes when the signal catalog is
deleted
     */
    public CompletableFuture<Void> deleteSignalCatalogAsync(String name) {
        DeleteSignalCatalogRequest request = DeleteSignalCatalogRequest.builder()
```

```
                .name(name)
                .build();

        return getAsyncClient().deleteSignalCatalog(request)
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }
                        throw new RuntimeException("Failed to delete the signal
catalog: " + cause);
                    }
                    logger.info("{} was successfully deleted", name);
                    return null;
                });
    }

    /**
     * Asynchronously retrieves a list of all nodes in the specified signal catalog.
     *
     * @param signalCatalogName the name of the signal catalog to retrieve nodes for
     * @return a {@link CompletableFuture} that, when completed, contains a {@link
List} of {@link Node} objects
     * representing all the nodes in the specified signal catalog
     */
    public CompletableFuture<List<Node>> listSignalCatalogNodeAsync(String
signalCatalogName) {
        ListSignalCatalogNodesRequest request =
ListSignalCatalogNodesRequest.builder()
                .name(signalCatalogName)
                .build();

        List<Node> allNodes = new ArrayList<>();

        return getAsyncClient().listSignalCatalogNodesPaginator(request)
                .subscribe(response -> allNodes.addAll(response.nodes()))
                .thenApply(v -> allNodes);
    }



    /**
```

```
    * Creates a model manifest.
    *
    * @param name             the name of the model manifest to create
    * @param signalCatalogArn the Amazon Resource Name (ARN) of the signal catalog
    * @param nodes            a list of nodes to include in the model manifest
    * @return a {@link CompletableFuture} that completes with the ARN of the
created model manifest
    */
   public CompletableFuture<String> createModelManifestAsync(String name,
                                                              String
signalCatalogArn,
                                                              List<Node> nodes) {
       // Extract the fully qualified names (FQNs) from each Node in the provided
list.
       List<String> fqnList = nodes.stream()
               .map(node -> {
                   if (node.sensor() != null) {
                       return node.sensor().fullyQualifiedName();
                   } else if (node.branch() != null) {
                       return node.branch().fullyQualifiedName();
                   } else if (node.attribute() != null) {
                       return node.attribute().fullyQualifiedName();
                   } else {
                       throw new RuntimeException("Unsupported node type");
                   }
               })
               .toList();

       CreateModelManifestRequest request = CreateModelManifestRequest.builder()
               .name(name)
               .signalCatalogArn(signalCatalogArn)
               .nodes(fqnList)
               .build();


       CompletableFuture<String> result = new CompletableFuture<>();
       getAsyncClient().createModelManifest(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                       if (cause instanceof InvalidSignalsException) {
```

```
                            result.completeExceptionally(new
CompletionException("The request contains signals that aren't valid: " +
cause.getMessage(), cause));
                        } else {
                            result.completeExceptionally(new
CompletionException("Failed to create model manifest: " + exception.getMessage(),
exception));
                        }
                    } else {
                        result.complete(response.arn()); // Complete successfully
with the ARN
                    }
                });

        return result;
    }


    /**
     * Deletes a fleet based on the provided fleet ID.
     *
     * @param fleetId the ID of the fleet to be deleted
     */
    public CompletableFuture<Void> deleteFleetAsync(String fleetId) {
        DeleteFleetRequest request = DeleteFleetRequest.builder()
                .fleetId(fleetId)
                .build();

        return getAsyncClient().deleteFleet(request)
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }
                        throw new RuntimeException("Failed to delete the fleet: " +
cause);
                    }
                    logger.info("{} was successfully deleted", fleetId);
                    return null;
                });
    }
```

```
    /**
     * Creates a new fleet.
     *
     * @param catARN  the Amazon Resource Name (ARN) of the signal catalog to
 associate with the fleet
     * @param fleetId the unique identifier for the fleet
     * @return a {@link CompletableFuture} that completes with the ID of the created
 fleet
     */
    public CompletableFuture<String> createFleetAsync(String catARN, String fleetId)
 {
        CreateFleetRequest fleetRequest = CreateFleetRequest.builder()
                .fleetId(fleetId)
                .signalCatalogArn(catARN)
                .description("Built using the AWS For Java V2")
                .build();

        CompletableFuture<String> result = new CompletableFuture<>();
        getAsyncClient().createFleet(fleetRequest)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
 exception.getCause() : exception;

                        if (cause instanceof ResourceNotFoundException) {
                            result.completeExceptionally(cause);
                        } else {
                            result.completeExceptionally(new RuntimeException("An
 unexpected error occurred", cause));
                        }
                    } else {
                        result.complete(response.id());
                    }
                });

        return result;
    }
}
```

操作

`createDecoderManifest`

以下代码示例演示了如何使用 `createDecoderManifest`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Creates a new decoder manifest.
     *
     * @param name             the name of the decoder manifest
     * @param modelManifestArn the ARN of the model manifest
     * @return a {@link CompletableFuture} that completes with the ARN of the
created decoder manifest
     */
    public CompletableFuture<String> createDecoderManifestAsync(String name, String
modelManifestArn) {
        String interfaceId = "can0";
        NetworkInterface networkInterface = NetworkInterface.builder()
                .interfaceId(interfaceId)
                .type(NetworkInterfaceType.CAN_INTERFACE)
                .canInterface(CanInterface.builder()
                        .name("canInterface0")
                        .protocolName("CAN")
                        .protocolVersion("1.0")
                        .build())
                .build();

        // Vehicle.Powertrain.EngineRPM decoder.
        SignalDecoder engineRpmDecoder = SignalDecoder.builder()
                .fullyQualifiedName("Vehicle.Powertrain.EngineRPM")
                .interfaceId(interfaceId)
                .type(SignalDecoderType.CAN_SIGNAL)
                .canSignal(CanSignal.builder()
                        .messageId(100)
```

```
                        .isBigEndian(false)
                        .isSigned(false)
                        .startBit(0)
                        .length(16)
                        .factor(1.0)
                        .offset(0.0)
                        .build())
                .build();

        // Vehicle.Powertrain.VehicleSpeed decoder.
        SignalDecoder vehicleSpeedDecoder = SignalDecoder.builder()
                .fullyQualifiedName("Vehicle.Powertrain.VehicleSpeed")
                .interfaceId(interfaceId)
                .type(SignalDecoderType.CAN_SIGNAL)
                .canSignal(CanSignal.builder()
                        .messageId(101)
                        .isBigEndian(false)
                        .isSigned(false)
                        .startBit(16)
                        .length(16)
                        .factor(1.0)
                        .offset(0.0)
                        .build())
                .build();

        CreateDecoderManifestRequest request =
CreateDecoderManifestRequest.builder()
                .name(name)
                .modelManifestArn(modelManifestArn)
                .networkInterfaces(List.of(networkInterface))
                .signalDecoders(List.of(engineRpmDecoder, vehicleSpeedDecoder))
                .build();

        CompletableFuture<String> result = new CompletableFuture<>();

        getAsyncClient().createDecoderManifest(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof DecoderManifestValidationException) {
```

```
                            result.completeExceptionally(new
CompletionException("The request contains signal decoders with validation errors: "
+ cause.getMessage(), cause));
                        } else {
                            result.completeExceptionally(new
CompletionException("Failed to create decoder manifest: " + exception.getMessage(),
exception));
                        }
                    } else {
                        result.complete(response.arn()); // Complete successfully
with the ARN
                    }
                });

        return result;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考createDecoderManifest中的。

`createFleet`

以下代码示例演示了如何使用 `createFleet`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Creates a new fleet.
     *
     * @param catARN  the Amazon Resource Name (ARN) of the signal catalog to
associate with the fleet
     * @param fleetId the unique identifier for the fleet
     * @return a {@link CompletableFuture} that completes with the ID of the created
fleet
     */
```

```
    public CompletableFuture<String> createFleetAsync(String catARN, String fleetId)
{
        CreateFleetRequest fleetRequest = CreateFleetRequest.builder()
                .fleetId(fleetId)
                .signalCatalogArn(catARN)
                .description("Built using the AWS For Java V2")
                .build();

        CompletableFuture<String> result = new CompletableFuture<>();
        getAsyncClient().createFleet(fleetRequest)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                        if (cause instanceof ResourceNotFoundException) {
                            result.completeExceptionally(cause);
                        } else {
                            result.completeExceptionally(new RuntimeException("An
unexpected error occurred", cause));
                        }
                    } else {
                        result.complete(response.id());
                    }
                });

        return result;
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 createFleet。

`createModelManifest`

以下代码示例演示了如何使用 `createModelManifest`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Creates a model manifest.
    *
    * @param name             the name of the model manifest to create
    * @param signalCatalogArn the Amazon Resource Name (ARN) of the signal catalog
    * @param nodes            a list of nodes to include in the model manifest
    * @return a {@link CompletableFuture} that completes with the ARN of the
created model manifest
    */
   public CompletableFuture<String> createModelManifestAsync(String name,
                                                              String
signalCatalogArn,
                                                              List<Node> nodes) {
       // Extract the fully qualified names (FQNs) from each Node in the provided
list.
       List<String> fqnList = nodes.stream()
               .map(node -> {
                   if (node.sensor() != null) {
                       return node.sensor().fullyQualifiedName();
                   } else if (node.branch() != null) {
                       return node.branch().fullyQualifiedName();
                   } else if (node.attribute() != null) {
                       return node.attribute().fullyQualifiedName();
                   } else {
                       throw new RuntimeException("Unsupported node type");
                   }
               })
               .toList();

       CreateModelManifestRequest request = CreateModelManifestRequest.builder()
               .name(name)
               .signalCatalogArn(signalCatalogArn)
               .nodes(fqnList)
               .build();


       CompletableFuture<String> result = new CompletableFuture<>();
       getAsyncClient().createModelManifest(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
```

```
                        if (cause instanceof InvalidSignalsException) {
                            result.completeExceptionally(new
CompletionException("The request contains signals that aren't valid: " +
cause.getMessage(), cause));
                        } else {
                            result.completeExceptionally(new
CompletionException("Failed to create model manifest: " + exception.getMessage(),
exception));
                        }
                    } else {
                        result.complete(response.arn()); // Complete successfully
with the ARN
                    }
                });

        return result;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[createModelManifest](#)中的。

`createSignalCatalog`

以下代码示例演示了如何使用 `createSignalCatalog`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Creates a signal catalog.
     *
     * @param signalCatalogName the name of the signal catalog to be created
     * @return a {@link CompletableFuture} that completes with the Amazon Resource
Name (ARN) of the created signal catalog
     */
```

```
   public CompletableFuture<String> createSignalCatalogAsync(String
signalCatalogName) {
       return deleteSignalCatalogIfExistsAsync(signalCatalogName)
               .thenCompose(ignored -> delayAsync(2000)) // Wait for 2 seconds
               .thenCompose(ignored -> {
                   List<Node> nodes = List.of(
                           Node.builder().branch(
                                   Branch.builder()
                                           .fullyQualifiedName("Vehicle")
                                           .description("Root branch")
                                           .build()
                           ).build(),
                           Node.builder().branch(
                                   Branch.builder()

.fullyQualifiedName("Vehicle.Powertrain")
                                           .description("Powertrain branch")
                                           .build()
                           ).build(),
                           Node.builder().sensor(
                                   Sensor.builder()

.fullyQualifiedName("Vehicle.Powertrain.EngineRPM")
                                           .description("Engine RPM")
                                           .dataType(NodeDataType.DOUBLE)
                                           .unit("rpm")
                                           .build()
                           ).build(),
                           Node.builder().sensor(
                                   Sensor.builder()

.fullyQualifiedName("Vehicle.Powertrain.VehicleSpeed")
                                           .description("Vehicle Speed")
                                           .dataType(NodeDataType.DOUBLE)
                                           .unit("km/h")
                                           .build()
                           ).build()
                   );

                   CreateSignalCatalogRequest request =
CreateSignalCatalogRequest.builder()
                           .name(signalCatalogName)
                           .nodes(nodes)
                           .build();
```

```
                    CompletableFuture<String> result = new CompletableFuture<>();

                    getAsyncClient().createSignalCatalog(request)
                            .whenComplete((response, exception) -> {
                                if (exception != null) {
                                    Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;

                                    if (cause instanceof ValidationException) {
                                        result.completeExceptionally(cause);
                                    } else {
                                        result.completeExceptionally(new
RuntimeException("Error creating the catalog", cause));
                                    }
                                } else {
                                    result.complete(response.arn());
                                }
                            });

                    return result;
                });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[createSignalCatalog](#)中的。

`createVehicle`

以下代码示例演示了如何使用 `createVehicle`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
```

```
    * Creates a new vehicle in the system.
    *
    * @param vecName     the name of the vehicle to be created
    * @param manifestArn the Amazon Resource Name (ARN) of the model manifest for
the vehicle
    * @param decArn      the Amazon Resource Name (ARN) of the decoder manifest for
the vehicle
    * @return a {@link CompletableFuture} that completes when the vehicle has been
created, or throws a
    */
   public CompletableFuture<Void> createVehicleAsync(String vecName, String
manifestArn, String decArn) {
       CreateVehicleRequest request = CreateVehicleRequest.builder()
               .vehicleName(vecName)
               .modelManifestArn(manifestArn)
               .decoderManifestArn(decArn)
               .build();

       CompletableFuture<Void> result = new CompletableFuture<>();
       getAsyncClient().createVehicle(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception instanceof CompletionException ?
exception.getCause() : exception;

                       if (cause instanceof ResourceNotFoundException) {
                           result.completeExceptionally(cause);
                       } else {
                           result.completeExceptionally(new
RuntimeException("Failed to create vehicle: " + cause.getMessage(), cause));
                       }
                   } else {
                       logger.info("Vehicle '{}' created successfully.", vecName);
                       result.complete(null); // mark future as complete
                   }
               });

       return result;
   }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 createVehicle。

deleteDecoderManifest

以下代码示例演示了如何使用 deleteDecoderManifest。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Deletes a decoder manifest.
     *
     * @param name the name of the decoder manifest to delete
     * @return a {@link CompletableFuture} that completes when the decoder manifest
has been deleted
     */
    public CompletableFuture<Void> deleteDecoderManifestAsync(String name) {
        return
getAsyncClient().deleteDecoderManifest(DeleteDecoderManifestRequest.builder().name(name).bu
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }
                        throw new RuntimeException("Failed to delete the decoder
manifest: " + cause);
                    }
                    return null;
                });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考deleteDecoderManifest中的。

deleteFleet

以下代码示例演示了如何使用 deleteFleet。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Deletes a fleet based on the provided fleet ID.
     *
     * @param fleetId the ID of the fleet to be deleted
     */
    public CompletableFuture<Void> deleteFleetAsync(String fleetId) {
        DeleteFleetRequest request = DeleteFleetRequest.builder()
                .fleetId(fleetId)
                .build();

        return getAsyncClient().deleteFleet(request)
                .handle((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }
                        throw new RuntimeException("Failed to delete the fleet: " +
cause);
                    }
                    logger.info("{} was successfully deleted", fleetId);
                    return null;
                });
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 deleteFleet。

`deleteModelManifest`

以下代码示例演示了如何使用 `deleteModelManifest`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Deletes a model manifest.
    *
    * @param name the name of the model manifest to delete
    * @return a {@link CompletableFuture} that completes when the model manifest
has been deleted
    */
   public CompletableFuture<Void> deleteModelManifestAsync(String name) {
       DeleteModelManifestRequest request = DeleteModelManifestRequest.builder()
               .name(name)
               .build();

       return getAsyncClient().deleteModelManifest(request)
               .handle((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                       if (cause instanceof ResourceNotFoundException) {
                           throw (ResourceNotFoundException) cause;
                       }
                       throw new RuntimeException("Failed to delete the model
manifest: " + cause);
                   }
                   logger.info("{} was successfully deleted", name);
                   return null;
               });
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考deleteModelManifest中的。

deleteSignalCatalog

以下代码示例演示了如何使用 deleteSignalCatalog。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Deletes a signal catalog.
    *
    * @param name the name of the signal catalog to delete
    * @return a {@link CompletableFuture} that completes when the signal catalog is
deleted
    */
   public CompletableFuture<Void> deleteSignalCatalogAsync(String name) {
       DeleteSignalCatalogRequest request = DeleteSignalCatalogRequest.builder()
               .name(name)
               .build();

       return getAsyncClient().deleteSignalCatalog(request)
               .handle((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                       if (cause instanceof ResourceNotFoundException) {
                           throw (ResourceNotFoundException) cause;
                       }
                       throw new RuntimeException("Failed to delete the signal
catalog: " + cause);
                   }
                   logger.info("{} was successfully deleted", name);
                   return null;
               });
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考deleteSignalCatalog中的。

deleteVehicle

以下代码示例演示了如何使用 deleteVehicle。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Deletes a vehicle with the specified name.
    *
    * @param vecName the name of the vehicle to be deleted
    * @return a {@link CompletableFuture} that completes when the vehicle has been
deleted
    */
   public CompletableFuture<Void> deleteVehicleAsync(String vecName) {
       DeleteVehicleRequest request = DeleteVehicleRequest.builder()
               .vehicleName(vecName)
               .build();

       return getAsyncClient().deleteVehicle(request)
               .handle((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
                       if (cause instanceof ResourceNotFoundException) {
                           throw (ResourceNotFoundException) cause;
                       }
                       throw new RuntimeException("Failed to delete the vehicle: "
+ cause);
                   }
                   return null;
               });
   }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 deleteVehicle。

`getDecoderManifest`

以下代码示例演示了如何使用 `getDecoderManifest`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Waits for the decoder manifest to become active.
    *
    * @param decoderName the name of the decoder to wait for
    * @return a {@link CompletableFuture} that completes when the decoder manifest
becomes active, or exceptionally if an error occurs or the manifest becomes invalid
    */
   public CompletableFuture<Void> waitForDecoderManifestActiveAsync(String
decoderName) {
       CompletableFuture<Void> result = new CompletableFuture<>();

       ScheduledExecutorService scheduler =
Executors.newSingleThreadScheduledExecutor();
       AtomicInteger secondsElapsed = new AtomicInteger(0);
       AtomicReference<ManifestStatus> lastStatus = new
AtomicReference<>(ManifestStatus.DRAFT);

       logger.info(" Elapsed: 0s | Decoder Status: DRAFT");

       final Runnable pollTask = new Runnable() {
           @Override
           public void run() {
               int elapsed = secondsElapsed.incrementAndGet();

               // Check status every 5 seconds
               if (elapsed % 5 == 0) {
```

```
                    GetDecoderManifestRequest request =
GetDecoderManifestRequest.builder()
                            .name(decoderName)
                            .build();

                    getAsyncClient().getDecoderManifest(request)
                            .whenComplete((response, exception) -> {
                                if (exception != null) {
                                    Throwable cause = exception instanceof
CompletionException ? exception.getCause() : exception;

                                    scheduler.shutdown();
                                    if (cause instanceof ResourceNotFoundException)
{
                                        result.completeExceptionally(new
RuntimeException("Decoder manifest not found: " + cause.getMessage(), cause));
                                    } else {
                                        result.completeExceptionally(new
RuntimeException("Error while polling decoder manifest status: " +
exception.getMessage(), exception));
                                    }
                                    return;
                                }

                                ManifestStatus status = response.status();
                                lastStatus.set(status);

                                if (status == ManifestStatus.ACTIVE) {
                                    logger.info("\r Elapsed: {}s | Decoder Status:
ACTIVE", elapsed);
                                    scheduler.shutdown();
                                    result.complete(null);
                                } else if (status == ManifestStatus.INVALID) {
                                    logger.info("\r Elapsed: {}s | Decoder Status:
INVALID", elapsed);
                                    scheduler.shutdown();
                                    result.completeExceptionally(new
RuntimeException("Decoder manifest became INVALID. Cannot proceed."));
                                } else {
                                    logger.info("\r# Elapsed: {}s | Decoder Status:
{}", elapsed, status);
                                }
                            });
                } else {
```

```
                    logger.info("\r Elapsed: {}s | Decoder Status: {}", elapsed,
lastStatus.get());
                }
            }
        };

        // Start the task with an initial delay of 1 second, and repeat every second
        scheduler.scheduleAtFixedRate(pollTask, 1, 1, TimeUnit.SECONDS);
        return result;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[getDecoderManifest](#)中的。

`getModelManifest`

以下代码示例演示了如何使用 `getModelManifest`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Waits for the specified model manifest to become active.
     *
     * @param manifestName the name of the model manifest to wait for
     */
    public CompletableFuture<Void> waitForModelManifestActiveAsync(String
manifestName) {
        CompletableFuture<Void> result = new CompletableFuture<>();

        ScheduledExecutorService scheduler =
Executors.newSingleThreadScheduledExecutor();
        AtomicInteger secondsElapsed = new AtomicInteger(0);
        AtomicReference<ManifestStatus> lastStatus = new
AtomicReference<>(ManifestStatus.DRAFT);
```

```
        logger.info("Elapsed: 0s | Status: DRAFT");

        final Runnable pollTask = new Runnable() {
            @Override
            public void run() {
                int elapsed = secondsElapsed.incrementAndGet();

                // Only check status every 5 seconds
                if (elapsed % 5 == 0) {
                    GetModelManifestRequest request =
GetModelManifestRequest.builder()
                            .name(manifestName)
                            .build();

                    getAsyncClient().getModelManifest(request)
                            .whenComplete((response, exception) -> {
                                if (exception != null) {
                                    Throwable cause = exception instanceof
CompletionException ? exception.getCause() : exception;

                                    scheduler.shutdown();
                                    if (cause instanceof ResourceNotFoundException)
{
                                        result.completeExceptionally(new
RuntimeException("Model manifest not found: " + cause.getMessage(), cause));
                                    } else {
                                        result.completeExceptionally(new
RuntimeException("Error while polling model manifest status: " +
exception.getMessage(), exception));
                                    }
                                    return;
                                }

                                ManifestStatus status = response.status();
                                lastStatus.set(status);

                                if (status == ManifestStatus.ACTIVE) {
                                    logger.info("\rElapsed: {}s | Status: ACTIVE",
elapsed);

                                    scheduler.shutdown();
                                    result.complete(null);
                                } else if (status == ManifestStatus.INVALID) {
```

```
                                    logger.info("\rElapsed: {}s | Status: INVALID",
elapsed);
                                    scheduler.shutdown();
                                    result.completeExceptionally(new
RuntimeException("Model manifest became INVALID. Cannot proceed."));
                                } else {
                                    logger.info("\rElapsed: {}s | Status: {}",
elapsed, status);
                                }
                            });
                } else {
                    logger.info("\rElapsed: {}s | Status: {}", elapsed,
lastStatus.get());
                }
            }
        };

        // Start the task with an initial delay of 1 second, and repeat every second
        scheduler.scheduleAtFixedRate(pollTask, 1, 1, TimeUnit.SECONDS);
        return result;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[getModelManifest](#)中的。

`getVehicle`

以下代码示例演示了如何使用 `getVehicle`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Fetches the details of a vehicle.
     *
```

```
    * @param vehicleName the name of the vehicle to fetch details for
    * @return a {@link CompletableFuture} that completes when the vehicle details
have been fetched
    */
   public CompletableFuture<Void> getVehicleDetailsAsync(String vehicleName) {
       GetVehicleRequest request = GetVehicleRequest.builder()
               .vehicleName(vehicleName)
               .build();

       CompletableFuture<Void> result = new CompletableFuture<>();

       getAsyncClient().getVehicle(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception instanceof CompletionException ?
exception.getCause() : exception;

                       if (cause instanceof ResourceNotFoundException) {
                           result.completeExceptionally(cause); // don't rewrap
                       } else {
                           result.completeExceptionally(new
RuntimeException("Failed to fetch vehicle details: " + cause.getMessage(), cause));
                       }
                   } else {
                       Map<String, Object> details = new HashMap<>();
                       details.put("vehicleName", response.vehicleName());
                       details.put("arn", response.arn());
                       details.put("modelManifestArn",
response.modelManifestArn());
                       details.put("decoderManifestArn",
response.decoderManifestArn());
                       details.put("attributes", response.attributes());
                       details.put("creationTime",
response.creationTime().toString());
                       details.put("lastModificationTime",
response.lastModificationTime().toString());

                       logger.info("Vehicle Details:");
                       details.forEach((key, value) -> logger.info("• {} : {}",
key, value));

                       result.complete(null); // mark as successful
                   }
               });
```

```
        return result;
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 getVehicle。

`listSignalCatalogNodes`

以下代码示例演示了如何使用 `listSignalCatalogNodes`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Asynchronously retrieves a list of all nodes in the specified signal catalog.
     *
     * @param signalCatalogName the name of the signal catalog to retrieve nodes for
     * @return a {@link CompletableFuture} that, when completed, contains a {@link
List} of {@link Node} objects
     * representing all the nodes in the specified signal catalog
     */
    public CompletableFuture<List<Node>> listSignalCatalogNodeAsync(String
signalCatalogName) {
        ListSignalCatalogNodesRequest request =
ListSignalCatalogNodesRequest.builder()
                .name(signalCatalogName)
                .build();

        List<Node> allNodes = new ArrayList<>();

        return getAsyncClient().listSignalCatalogNodesPaginator(request)
                .subscribe(response -> allNodes.addAll(response.nodes()))
                .thenApply(v -> allNodes);
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考中的listSignalCatalog节点。

`updateDecoderManifest`

以下代码示例演示了如何使用 `updateDecoderManifest`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Updates the decoder manifest with the given name.
    *
    * @param name the name of the decoder manifest to update
    * @return a {@link CompletableFuture} that completes when the update operation
is finished
    */
   public CompletableFuture<Void> updateDecoderManifestAsync(String name) {
       UpdateDecoderManifestRequest request =
UpdateDecoderManifestRequest.builder()
               .name(name)
               .status(ManifestStatus.ACTIVE)
               .build();

       return getAsyncClient().updateDecoderManifest(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       throw new CompletionException("Failed to update decoder
manifest: " + exception.getMessage(), exception);
                   }
               })
               .thenApply(response -> null);
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考updateDecoderManifest中的。

`updateModelManifest`

以下代码示例演示了如何使用 `updateModelManifest`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Updates the model manifest.
     *
     * @param name the name of the model manifest to update
     */
    public void updateModelManifestAsync(String name) {
        UpdateModelManifestRequest request = UpdateModelManifestRequest.builder()
                .name(name)
                .status(ManifestStatus.ACTIVE)
                .build();

        getAsyncClient().updateModelManifest(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        throw new CompletionException("Failed to update model
manifest: " + exception.getMessage(), exception);
                    }
                })
                .thenApply(response -> null);
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考updateModelManifest中的。

AWS IoT SiteWise 使用适用于 Java 的 SDK 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 SiteWise。 AWS SDK for Java 2.x

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 基本功能
- 操作

开始使用

你好 AWS IoT SiteWise

以下代码示例展示了如何开始使用 AWS IoT SiteWise。

适用于 Java 的 SDK 2.x

> **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。

```
public class HelloSitewise {
    private static final Logger logger =
 LoggerFactory.getLogger(HelloSitewise.class);
    public static void main(String[] args) {
         fetchAssetModels();
    }

    /**
```

```
     * Fetches asset models using the provided {@link IoTSiteWiseAsyncClient}.
     */
    public static void fetchAssetModels() {
        IoTSiteWiseAsyncClient siteWiseAsyncClient =
 IoTSiteWiseAsyncClient.create();
        ListAssetModelsRequest assetModelsRequest = ListAssetModelsRequest.builder()
            .assetModelTypes(AssetModelType.ASSET_MODEL)
            .build();

        // Asynchronous paginator - process paginated results.
        ListAssetModelsPublisher listModelsPaginator =
 siteWiseAsyncClient.listAssetModelsPaginator(assetModelsRequest);
        CompletableFuture<Void> future = listModelsPaginator.subscribe(response -> {
            response.assetModelSummaries().forEach(assetSummary ->
                logger.info("Asset Model Name: {} ", assetSummary.name())
            );
        });

        // Wait for the asynchronous operation to complete
        future.join();
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListAssetModels中的。

基本功能

了解基本功能

以下代码示例展示了如何：

- 创建 AWS IoT SiteWise 资产模型。
- 创建 AWS IoT SiteWise 资产。
- 检索属性 ID 值。
- 向 AWS IoT SiteWise 资产发送数据。
- 检索 AWS IoT SiteWise 资产属性的值。
- 创建 AWS IoT SiteWise 门户。
- 创建 AWS IoT SiteWise 网关。
- 描述网 AWS IoT SiteWise 关。

- 删除资 AWS IoT SiteWise 产。

适用于 Java 的 SDK 2.x

> **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。

运行演示 AWS IoT SiteWise 功能的交互式场景。

```
public class SitewiseScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    private static final Logger logger =
 LoggerFactory.getLogger(SitewiseScenario.class);
    static Scanner scanner = new Scanner(System.in);

    private static final String ROLES_STACK = "RoleSitewise";

    static SitewiseActions sitewiseActions = new SitewiseActions();

    public static void main(String[] args) throws Throwable {
        Scanner scanner = new Scanner(System.in);
        String contactEmail = "user@mydomain.com"; // Change email address.
        String assetModelName = "MyAssetModel1";
        String assetName = "MyAsset1" ;
        String portalName = "MyPortal1" ;
        String gatewayName = "MyGateway1" ;
        String myThing =  "MyThing1" ;

        logger.info("""
            AWS IoT SiteWise is a fully managed software-as-a-service (SaaS) that
            makes it easy to collect, store, organize, and monitor data from
 industrial equipment and processes.
            It is designed to help industrial and manufacturing organizations
 collect data from their equipment and
            processes, and use that data to make informed decisions about their
 operations.
```

```
            One of the key features of AWS IoT SiteWise is its ability to connect to
a wide range of industrial
            equipment and systems, including programmable logic controllers (PLCs),
sensors, and other
            industrial devices. It can collect data from these devices and organize
it into a unified data model,
            making it easier to analyze and gain insights from the data. AWS IoT
SiteWise also provides tools for
            visualizing the data, setting up alarms and alerts, and generating
reports.

            Another key feature of AWS IoT SiteWise is its ability to scale to
handle large volumes of data.
            It can collect and store data from thousands of devices and process
millions of data points per second,
            making it suitable for large-scale industrial operations. Additionally,
AWS IoT SiteWise is designed
            to be secure and compliant, with features like role-based access
controls, data encryption,
            and integration with other AWS services for additional security and
compliance features.

            Let's get started...
            """);

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        try {
            runScenario(assetModelName, assetName, portalName, contactEmail,
gatewayName, myThing);
        } catch (RuntimeException e) {
           logger.info(e.getMessage());
        }
    }

    public static void runScenario(String assetModelName, String assetName,  String
portalName, String contactEmail, String gatewayName, String myThing) throws
Throwable {
        logger.info("Use AWS CloudFormation to create an IAM role that is required
for this scenario.");
        CloudFormationHelper.deployCloudFormationStack(ROLES_STACK);
        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputsAsync(ROLES_STACK).join();
```

```
        String iamRole = stackOutputs.get("SitewiseRoleArn");
        logger.info("The ARN of the IAM role is {}",iamRole);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("1. Create an AWS SiteWise Asset Model");
        logger.info("""
             An AWS IoT SiteWise Asset Model is a way to represent the physical
assets, such as equipment,
             processes, and systems, that exist in an industrial environment. This
model provides a structured and
             hierarchical representation of these assets, allowing users to define
the relationships and properties
             of each asset.

             This scenario creates two asset model properties: temperature and
humidity.
            """);
        waitForInputToContinue(scanner);
        String assetModelId = null;
        try {
            CreateAssetModelResponse response =
sitewiseActions.createAssetModelAsync(assetModelName).join();
            assetModelId = response.assetModelId();
            logger.info("Asset Model successfully created. Asset Model ID: {}. ",
assetModelId);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceAlreadyExistsException) {
                try {
                    assetModelId =
sitewiseActions.getAssetModelIdAsync(assetModelName).join();
                    logger.info("The Asset Model {} already exists. The id of the
existing model is {}. Moving on...", assetModelName, assetModelId);
                } catch (CompletionException cex) {
                    logger.error("Exception thrown acquiring the asset model id:
{}", cex.getCause().getCause(), cex);
                    return;
                }
            } else {
                logger.info("An unexpected error occurred: " + cause.getMessage(),
cause);
                return;
            }
```

```
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("2. Create an AWS IoT SiteWise Asset");
        logger.info("""
             The IoT SiteWise model that we just created defines the structure and
metadata for your physical assets.
             Now we create an asset from the asset model.

            """);
        logger.info("Let's wait 30 seconds for the asset to be ready.");
        countdown(30);
        waitForInputToContinue(scanner);
        String assetId;
        try {
            CreateAssetResponse response =
sitewiseActions.createAssetAsync(assetName, assetModelId).join();
            assetId = response.assetId();
            logger.info("Asset created with ID: {}", assetId);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The asset model id was not found: {}",
cause.getMessage(), cause);
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Retrieve the property ID values");
        logger.info("""
             To send data to an asset, we need to get the property ID values. In
this scenario, we access the
             temperature and humidity property ID values.
            """);
        waitForInputToContinue(scanner);
        Map<String, String>  propertyIds = null;
        try {
```

```
            propertyIds = sitewiseActions.getPropertyIds(assetModelId).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IoTSiteWiseException) {
                logger.error("IoTSiteWiseException occurred: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
            return;
        }
        String humPropId =  propertyIds.get("Humidity");
        logger.info("The Humidity property Id is {}", humPropId);
        String tempPropId = propertyIds.get("Temperature");
        logger.info("The Temperature property Id is {}", tempPropId);

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Send data to an AWS IoT SiteWise Asset");
        logger.info("""
            By sending data to an IoT SiteWise Asset, you can aggregate data from
            multiple sources, normalize the data into a standard format, and store
it in a
            centralized location. This makes it easier to analyze and gain insights
from the data.

            In this example, we generate sample temperature and humidity data and
send it to the AWS IoT SiteWise asset.

            """);
        waitForInputToContinue(scanner);
        try {
            sitewiseActions.sendDataToSiteWiseAsync(assetId, tempPropId,
humPropId).join();
            logger.info("Data sent successfully.");
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The AWS resource was not found: {}",
cause.getMessage(), cause);
            } else {
```

```
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Retrieve the value of the IoT SiteWise Asset property");
        logger.info("""
            IoT SiteWise is an AWS service that allows you to collect, process, and
analyze industrial data
            from connected equipment and sensors. One of the key benefits of reading
an IoT SiteWise property
            is the ability to gain valuable insights from your industrial data.

            """);
        waitForInputToContinue(scanner);
        try {
            Double assetVal = sitewiseActions.getAssetPropValueAsync(tempPropId,
assetId).join();
            logger.info("The property name is: {}", "Temperature");
            logger.info("The value of this property is: {}", assetVal);

            waitForInputToContinue(scanner);

            assetVal = sitewiseActions.getAssetPropValueAsync(humPropId,
assetId).join();
            logger.info("The property name is: {}", "Humidity");
            logger.info("The value of this property is: {}", assetVal);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    logger.info("The AWS resource was not found: {}",
cause.getMessage(), cause);
                } else {
                    logger.info("An unexpected error occurred: {}",
cause.getMessage(), cause);
                }
                return;
            }
        waitForInputToContinue(scanner);
        logger.info(DASHES);
```

```
        logger.info(DASHES);
        logger.info("6. Create an IoT SiteWise Gateway");
        logger.info(
            """
                IoT SiteWise Gateway serves as the bridge between industrial
equipment, sensors, and the
                cloud-based IoT SiteWise service. It is responsible for securely
collecting, processing, and
                transmitting data from various industrial assets to the IoT SiteWise
platform,
                enabling real-time monitoring, analysis, and optimization of
industrial operations.

                """);
        waitForInputToContinue(scanner);
        String gatewayId = "";
        try {
            gatewayId = sitewiseActions.createGatewayAsync(gatewayName,
myThing).join();
            logger.info("Gateway creation completed successfully. id is {}",
gatewayId );
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IoTSiteWiseException siteWiseEx) {
                logger.error("IoT SiteWise error occurred: Error message: {}, Error
code {}",
                        siteWiseEx.getMessage(),
siteWiseEx.awsErrorDetails().errorCode(), siteWiseEx);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage());
            }
            return;
        }
        logger.info(DASHES);
        logger.info(DASHES);

        logger.info("7. Describe the IoT SiteWise Gateway");
         waitForInputToContinue(scanner);
        try {
            sitewiseActions.describeGatewayAsync(gatewayId)
                .thenAccept(response -> {
                    logger.info("Gateway Name: {}", response.gatewayName());
```

```
                    logger.info("Gateway ARN: {}", response.gatewayArn());
                    logger.info("Gateway Platform: {}", response.gatewayPlatform());
                    logger.info("Gateway Creation Date: {}",
response.creationDate());
                }).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException notFoundException) {
                logger.error("A ResourceNotFoundException occurred: Error message:
{}, Error code {}",
                        notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Delete the AWS IoT SiteWise Assets");
        logger.info(
            """
            Before you can delete the Asset Model, you must delete the assets.

            """);
        logger.info("Would you like to delete the IoT SiteWise Assets? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete the SiteWise assets.");

            try {
                sitewiseActions.deleteGatewayAsync(gatewayId).join();
                logger.info("Gateway {} was deleted successfully.", gatewayId);
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException notFoundException) {
                    logger.error("A ResourceNotFoundException occurred: Error
message: {}, Error code {}",
                            notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
                } else {
```

```
                    logger.error("An unexpected error occurred: {}",
cause.getMessage());
                }
            }

            try {
                sitewiseActions.deleteAssetAsync(assetId).join();
                logger.info("Request to delete asset {} sent successfully",
assetId);
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException notFoundException) {
                    logger.error("A ResourceNotFoundException occurred: Error
message: {}, Error code {}",
                            notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage());
                }
            }
            logger.info("Let's wait 1 minute for the asset to be deleted.");
            countdown(60);
            waitForInputToContinue(scanner);
            logger.info("Delete the AWS IoT SiteWise Asset Model");
            try {
                sitewiseActions.deleteAssetModelAsync(assetModelId).join();
                logger.info("Asset model deleted successfully.");
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException notFoundException) {
                    logger.error("A ResourceNotFoundException occurred: Error
message: {}, Error code {}",
                            notFoundException.getMessage(),
notFoundException.awsErrorDetails().errorCode(), notFoundException);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage());
                }
            }
            waitForInputToContinue(scanner);

        } else {
            logger.info("The resources will not be deleted.");
```

```
        }
        logger.info(DASHES);

        logger.info(DASHES);
        CloudFormationHelper.destroyCloudFormationStack(ROLES_STACK);
        logger.info("This concludes the AWS IoT SiteWise Scenario");
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }

    public static void countdown(int totalSeconds) throws InterruptedException {
        for (int i = totalSeconds; i >= 0; i--) {
            int displayMinutes = i / 60;
            int displaySeconds = i % 60;
            System.out.printf("\r%02d:%02d", displayMinutes, displaySeconds);
            Thread.sleep(1000); // Wait for 1 second
        }
        System.out.println(); // Move to the next line after countdown
        logger.info("Countdown complete!");
    }
}
```

AWS IoT SiteWise SDK 方法的包装器类。

```
public class SitewiseActions {
```

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

   private static IoTSiteWiseAsyncClient ioTSiteWiseAsyncClient;

   private static IoTSiteWiseAsyncClient getAsyncClient() {
       if (ioTSiteWiseAsyncClient == null) {
           SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
               .maxConcurrency(100)
               .connectionTimeout(Duration.ofSeconds(60))
               .readTimeout(Duration.ofSeconds(60))
               .writeTimeout(Duration.ofSeconds(60))
               .build();

           ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
               .apiCallTimeout(Duration.ofMinutes(2))
               .apiCallAttemptTimeout(Duration.ofSeconds(90))
               .retryStrategy(RetryMode.STANDARD)
               .build();

           ioTSiteWiseAsyncClient = IoTSiteWiseAsyncClient.builder()
               .httpClient(httpClient)
               .overrideConfiguration(overrideConfig)
               .build();
       }
       return ioTSiteWiseAsyncClient;
   }


   /**
    * Creates an asset model.
    *
    * @param name the name of the asset model to create.
    * @return a {@link CompletableFuture} that represents a {@link
CreateAssetModelResponse} result. The calling code
    *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
    *         available to the calling code as a {@link CompletionException}. By
calling
```

```
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<CreateAssetModelResponse> createAssetModelAsync(String
name) {
        PropertyType humidity = PropertyType.builder()
            .measurement(Measurement.builder().build())
            .build();

        PropertyType temperaturePropertyType = PropertyType.builder()
            .measurement(Measurement.builder().build())
            .build();

        AssetModelPropertyDefinition temperatureProperty =
AssetModelPropertyDefinition.builder()
            .name("Temperature")
            .dataType(PropertyDataType.DOUBLE)
            .type(temperaturePropertyType)
            .build();

        AssetModelPropertyDefinition humidityProperty =
AssetModelPropertyDefinition.builder()
            .name("Humidity")
            .dataType(PropertyDataType.DOUBLE)
            .type(humidity)
            .build();

        CreateAssetModelRequest createAssetModelRequest =
CreateAssetModelRequest.builder()
            .assetModelName(name)
            .assetModelDescription("This is my asset model")
            .assetModelProperties(temperatureProperty, humidityProperty)
            .build();

        return getAsyncClient().createAssetModel(createAssetModelRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to create asset model: {} ",
exception.getCause().getMessage());
                }
            });
    }
```

```
   /**
    * Creates an asset with the specified name and asset model Id.
    *
    * @param assetName    the name of the asset to create.
    * @param assetModelId the Id of the asset model to associate with the asset.
    * @return a {@link CompletableFuture} that represents a {@link
CreateAssetResponse} result. The calling code can
    *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
    *         available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<CreateAssetResponse> createAssetAsync(String assetName,
String assetModelId) {
       CreateAssetRequest createAssetRequest = CreateAssetRequest.builder()
           .assetModelId(assetModelId)
           .assetDescription("Created using the AWS SDK for Java")
           .assetName(assetName)
           .build();

       return getAsyncClient().createAsset(createAssetRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("Failed to create asset: {}",
exception.getCause().getMessage());
               }
           });
   }

   /**
    * Sends data to the SiteWise service.
    *
    * @param assetId        the ID of the asset to which the data will be sent.
    * @param tempPropertyId the ID of the temperature property.
    * @param humidityPropId the ID of the humidity property.
    * @return a {@link CompletableFuture} that represents a {@link
BatchPutAssetPropertyValueResponse} result. The
```

```
    *         calling code can attach callbacks, then handle the result or
exception by calling
    *         {@link CompletableFuture#join()} or {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
    *         available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<BatchPutAssetPropertyValueResponse>
sendDataToSiteWiseAsync(String assetId, String tempPropertyId, String
humidityPropId) {
       Map<String, Double> sampleData = generateSampleData();
       long timestamp = Instant.now().toEpochMilli();

       TimeInNanos time = TimeInNanos.builder()
           .timeInSeconds(timestamp / 1000)
           .offsetInNanos((int) ((timestamp % 1000) * 1000000))
           .build();

       BatchPutAssetPropertyValueRequest request =
BatchPutAssetPropertyValueRequest.builder()
           .entries(Arrays.asList(
               PutAssetPropertyValueEntry.builder()
                   .entryId("entry-3")
                   .assetId(assetId)
                   .propertyId(tempPropertyId)
                   .propertyValues(Arrays.asList(
                       AssetPropertyValue.builder()
                           .value(Variant.builder()
                               .doubleValue(sampleData.get("Temperature"))
                               .build())
                           .timestamp(time)
                           .build()
                   ))
                   .build(),
               PutAssetPropertyValueEntry.builder()
                   .entryId("entry-4")
                   .assetId(assetId)
                   .propertyId(humidityPropId)
                   .propertyValues(Arrays.asList(
                       AssetPropertyValue.builder()
```

```
                            .value(Variant.builder()
                                .doubleValue(sampleData.get("Humidity"))
                                .build())
                            .timestamp(time)
                            .build()
                    ))
                    .build()
            ))
            .build();

        return getAsyncClient().batchPutAssetPropertyValue(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An exception occurred: {}",
exception.getCause().getMessage());
                }
            });
    }

    /**
     * Fetches the value of an asset property.
     *
     * @param propId  the ID of the asset property to fetch.
     * @param assetId the ID of the asset to fetch the property value for.
     * @return a {@link CompletableFuture} that represents a {@link Double} result.
The calling code can attach
     *         callbacks, then handle the result or exception by calling {@link
CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<Double> getAssetPropValueAsync(String propId, String
assetId) {
        GetAssetPropertyValueRequest assetPropertyValueRequest =
GetAssetPropertyValueRequest.builder()
                .propertyId(propId)
                .assetId(assetId)
                .build();
```

```
       return getAsyncClient().getAssetPropertyValue(assetPropertyValueRequest)
               .handle((response, exception) -> {
                   if (exception != null) {
                       logger.error("Error occurred while fetching property value:
{}.", exception.getCause().getMessage());
                       throw (CompletionException) exception;
                   }
                   return response.propertyValue().value().doubleValue();
               });
   }

   /**
    * Retrieves the property IDs associated with a specific asset model.
    *
    * @param assetModelId the ID of the asset model that defines the properties.
    * @return a {@link CompletableFuture} that represents a {@link Map} result that
associates the property name to the
    *         propert ID. The calling code can attach callbacks, then handle the
result or exception by calling
    *         {@link CompletableFuture#join()} or {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<Map<String, String>> getPropertyIds(String
assetModelId) {
       ListAssetModelPropertiesRequest modelPropertiesRequest =
ListAssetModelPropertiesRequest.builder().assetModelId(assetModelId).build();
       return getAsyncClient().listAssetModelProperties(modelPropertiesRequest)
           .handle((response, throwable) -> {
               if (response != null) {
                   return response.assetModelPropertySummaries().stream()
                       .collect(Collectors
                           .toMap(AssetModelPropertySummary::name,
AssetModelPropertySummary::id));
               } else {
                   logger.error("Error occurred while fetching property IDs: {}.",
throwable.getCause().getMessage());
                   throw (CompletionException) throwable;
```

```
                }
            });
    }

    /**
     * Deletes an asset.
     *
     * @param assetId the ID of the asset to be deleted.
     * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetResponse} result. The calling code can
     *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<DeleteAssetResponse> deleteAssetAsync(String assetId) {
        DeleteAssetRequest deleteAssetRequest = DeleteAssetRequest.builder()
            .assetId(assetId)
            .build();

        return getAsyncClient().deleteAsset(deleteAssetRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An error occurred deleting asset with id: {}",
assetId);
                }
            });
    }

    /**
     * Deletes an Asset Model with the specified ID.
     *
     * @param assetModelId the ID of the Asset Model to delete.
     * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetModelResponse} result. The calling code
     *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
```

```
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<DeleteAssetModelResponse> deleteAssetModelAsync(String
assetModelId) {
       DeleteAssetModelRequest deleteAssetModelRequest =
DeleteAssetModelRequest.builder()
           .assetModelId(assetModelId)
           .build();

       return getAsyncClient().deleteAssetModel(deleteAssetModelRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("Failed to delete asset model with ID:{}.",
exception.getMessage());
               }
           });
   }

   /**
    * Retrieves the asset model ID for the given asset model name.
    *
    * @param assetModelName the name of the asset model for the ID.
    * @return a {@link CompletableFuture} that represents a {@link String} result
of the asset model ID or null if the
    *         asset model cannot be found. The calling code can attach callbacks,
then handle the result or exception
    *         by calling {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<String> getAssetModelIdAsync(String assetModelName) {
```

```
        ListAssetModelsRequest listAssetModelsRequest =
ListAssetModelsRequest.builder().build();
        return getAsyncClient().listAssetModels(listAssetModelsRequest)
                .handle((listAssetModelsResponse, exception) -> {
                    if (exception != null) {
                        logger.error("Failed to retrieve Asset Model ID: {}",
exception.getCause().getMessage());
                        throw (CompletionException) exception;
                    }
                    for (AssetModelSummary assetModelSummary :
listAssetModelsResponse.assetModelSummaries()) {
                        if (assetModelSummary.name().equals(assetModelName)) {
                            return assetModelSummary.id();
                        }
                    }
                    return null;
                });
    }


    /**
     * Creates a new IoT Sitewise gateway.
     *
     * @param gatewayName The name of the gateway to create.
     * @param myThing     The name of the core device thing to associate with the
gateway.
     * @return a {@link CompletableFuture} that represents a {@link String} result
of the gateways ID. The calling code
     *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<String> createGatewayAsync(String gatewayName, String
myThing) {
        GreengrassV2 gg = GreengrassV2.builder()
            .coreDeviceThingName(myThing)
            .build();
```

```
        GatewayPlatform platform = GatewayPlatform.builder()
            .greengrassV2(gg)
            .build();

        Map<String, String> tag = new HashMap<>();
        tag.put("Environment", "Production");

        CreateGatewayRequest createGatewayRequest = CreateGatewayRequest.builder()
            .gatewayName(gatewayName)
            .gatewayPlatform(platform)
            .tags(tag)
            .build();

        return getAsyncClient().createGateway(createGatewayRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    logger.error("Error creating the gateway.");
                    throw (CompletionException) exception;
                }
                logger.info("The ARN of the gateway is {}" ,
response.gatewayArn());
                return response.gatewayId();
            });
    }

    /**
     * Deletes the specified gateway.
     *
     * @param gatewayId the ID of the gateway to delete.
     * @return a {@link CompletableFuture} that represents a {@link
DeleteGatewayResponse} result.. The calling code
     *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
```

```
   public CompletableFuture<DeleteGatewayResponse> deleteGatewayAsync(String
gatewayId) {
       DeleteGatewayRequest deleteGatewayRequest = DeleteGatewayRequest.builder()
           .gatewayId(gatewayId)
           .build();

       return getAsyncClient().deleteGateway(deleteGatewayRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("Failed to delete gateway: {}",
exception.getCause().getMessage());
               }
           });
   }

   /**
    * Describes the specified gateway.
    *
    * @param gatewayId the ID of the gateway to describe.
    * @return a {@link CompletableFuture} that represents a {@link
DescribeGatewayResponse} result. The calling code
    *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<DescribeGatewayResponse> describeGatewayAsync(String
gatewayId) {
       DescribeGatewayRequest request = DescribeGatewayRequest.builder()
           .gatewayId(gatewayId)
           .build();

       return getAsyncClient().describeGateway(request)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("An error occurred during the describeGateway
method: {}", exception.getCause().getMessage());
               }
```

```
            });
    }

    private static Map<String, Double> generateSampleData() {
        Map<String, Double> data = new HashMap<>();
        data.put("Temperature", 23.5);
        data.put("Humidity", 65.0);
        return data;
    }
}
```

操作

`BatchPutAssetPropertyValue`

以下代码示例演示了如何使用 `BatchPutAssetPropertyValue`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Sends data to the SiteWise service.
     *
     * @param assetId        the ID of the asset to which the data will be sent.
     * @param tempPropertyId the ID of the temperature property.
     * @param humidityPropId the ID of the humidity property.
     * @return a {@link CompletableFuture} that represents a {@link
BatchPutAssetPropertyValueResponse} result. The
     *         calling code can attach callbacks, then handle the result or
exception by calling
     *         {@link CompletableFuture#join()} or {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
```

```
    *         available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<BatchPutAssetPropertyValueResponse>
sendDataToSiteWiseAsync(String assetId, String tempPropertyId, String
humidityPropId) {
       Map<String, Double> sampleData = generateSampleData();
       long timestamp = Instant.now().toEpochMilli();

       TimeInNanos time = TimeInNanos.builder()
           .timeInSeconds(timestamp / 1000)
           .offsetInNanos((int) ((timestamp % 1000) * 1000000))
           .build();

       BatchPutAssetPropertyValueRequest request =
BatchPutAssetPropertyValueRequest.builder()
           .entries(Arrays.asList(
               PutAssetPropertyValueEntry.builder()
                   .entryId("entry-3")
                   .assetId(assetId)
                   .propertyId(tempPropertyId)
                   .propertyValues(Arrays.asList(
                       AssetPropertyValue.builder()
                           .value(Variant.builder()
                               .doubleValue(sampleData.get("Temperature"))
                               .build())
                           .timestamp(time)
                           .build()
                   ))
                   .build(),
               PutAssetPropertyValueEntry.builder()
                   .entryId("entry-4")
                   .assetId(assetId)
                   .propertyId(humidityPropId)
                   .propertyValues(Arrays.asList(
                       AssetPropertyValue.builder()
                           .value(Variant.builder()
                               .doubleValue(sampleData.get("Humidity"))
                               .build())
                           .timestamp(time)
                           .build()
                   ))
```

```
                    .build()
            ))
            .build();

        return getAsyncClient().batchPutAssetPropertyValue(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An exception occurred: {}",
exception.getCause().getMessage());
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[BatchPutAssetPropertyValue]中的。

`CreateAsset`

以下代码示例演示了如何使用 `CreateAsset`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Creates an asset with the specified name and asset model Id.
     *
     * @param assetName    the name of the asset to create.
     * @param assetModelId the Id of the asset model to associate with the asset.
     * @return a {@link CompletableFuture} that represents a {@link
CreateAssetResponse} result. The calling code can
     *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
```

```
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
     *         available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<CreateAssetResponse> createAssetAsync(String assetName,
String assetModelId) {
        CreateAssetRequest createAssetRequest = CreateAssetRequest.builder()
            .assetModelId(assetModelId)
            .assetDescription("Created using the AWS SDK for Java")
            .assetName(assetName)
            .build();

        return getAsyncClient().createAsset(createAssetRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to create asset: {}",
exception.getCause().getMessage());
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateAsset中的。

`CreateAssetModel`

以下代码示例演示了如何使用 `CreateAssetModel`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Creates an asset model.
```

```
    *
    * @param name the name of the asset model to create.
    * @return a {@link CompletableFuture} that represents a {@link
CreateAssetModelResponse} result. The calling code
    *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps it
    *         available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<CreateAssetModelResponse> createAssetModelAsync(String
name) {
       PropertyType humidity = PropertyType.builder()
           .measurement(Measurement.builder().build())
           .build();

       PropertyType temperaturePropertyType = PropertyType.builder()
           .measurement(Measurement.builder().build())
           .build();

       AssetModelPropertyDefinition temperatureProperty =
AssetModelPropertyDefinition.builder()
           .name("Temperature")
           .dataType(PropertyDataType.DOUBLE)
           .type(temperaturePropertyType)
           .build();

       AssetModelPropertyDefinition humidityProperty =
AssetModelPropertyDefinition.builder()
           .name("Humidity")
           .dataType(PropertyDataType.DOUBLE)
           .type(humidity)
           .build();

       CreateAssetModelRequest createAssetModelRequest =
CreateAssetModelRequest.builder()
           .assetModelName(name)
           .assetModelDescription("This is my asset model")
           .assetModelProperties(temperatureProperty, humidityProperty)
```

```
            .build();

        return getAsyncClient().createAssetModel(createAssetModelRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("Failed to create asset model: {} ",
exception.getCause().getMessage());
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateAssetModel中的。

`CreateGateway`

以下代码示例演示了如何使用 `CreateGateway`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Creates a new IoT Sitewise gateway.
     *
     * @param gatewayName The name of the gateway to create.
     * @param myThing     The name of the core device thing to associate with the
gateway.
     * @return a {@link CompletableFuture} that represents a {@link String} result
of the gateways ID. The calling code
     *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
```

```
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<String> createGatewayAsync(String gatewayName, String
myThing) {
        GreengrassV2 gg = GreengrassV2.builder()
            .coreDeviceThingName(myThing)
            .build();

        GatewayPlatform platform = GatewayPlatform.builder()
            .greengrassV2(gg)
            .build();

        Map<String, String> tag = new HashMap<>();
        tag.put("Environment", "Production");

        CreateGatewayRequest createGatewayRequest = CreateGatewayRequest.builder()
            .gatewayName(gatewayName)
            .gatewayPlatform(platform)
            .tags(tag)
            .build();

        return getAsyncClient().createGateway(createGatewayRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    logger.error("Error creating the gateway.");
                    throw (CompletionException) exception;
                }
                logger.info("The ARN of the gateway is {}" ,
response.gatewayArn());
                return response.gatewayId();
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateGateway中的。

DeleteAsset

以下代码示例演示了如何使用 DeleteAsset。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Deletes an asset.
    *
    * @param assetId the ID of the asset to be deleted.
    * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetResponse} result. The calling code can
    *         attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<DeleteAssetResponse> deleteAssetAsync(String assetId) {
       DeleteAssetRequest deleteAssetRequest = DeleteAssetRequest.builder()
           .assetId(assetId)
           .build();

       return getAsyncClient().deleteAsset(deleteAssetRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("An error occurred deleting asset with id: {}",
assetId);
               }
           });
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteAsset中的。

DeleteAssetModel

以下代码示例演示了如何使用 DeleteAssetModel。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Deletes an Asset Model with the specified ID.
    *
    * @param assetModelId the ID of the Asset Model to delete.
    * @return a {@link CompletableFuture} that represents a {@link
DeleteAssetModelResponse} result. The calling code
    *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<DeleteAssetModelResponse> deleteAssetModelAsync(String
assetModelId) {
       DeleteAssetModelRequest deleteAssetModelRequest =
DeleteAssetModelRequest.builder()
           .assetModelId(assetModelId)
           .build();

       return getAsyncClient().deleteAssetModel(deleteAssetModelRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   logger.error("Failed to delete asset model with ID:{}.",
exception.getMessage());
               }
           });
```

```
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteAssetModel中的。

`DeleteGateway`

以下代码示例演示了如何使用 `DeleteGateway`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Deletes the specified gateway.
    *
    * @param gatewayId the ID of the gateway to delete.
    * @return a {@link CompletableFuture} that represents a {@link
DeleteGatewayResponse} result.. The calling code
    *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<DeleteGatewayResponse> deleteGatewayAsync(String
gatewayId) {
       DeleteGatewayRequest deleteGatewayRequest = DeleteGatewayRequest.builder()
           .gatewayId(gatewayId)
           .build();

       return getAsyncClient().deleteGateway(deleteGatewayRequest)
           .whenComplete((response, exception) -> {
```

```
                if (exception != null) {
                    logger.error("Failed to delete gateway: {}",
exception.getCause().getMessage());
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteGateway中的。

`DescribeAssetModel`

以下代码示例演示了如何使用 `DescribeAssetModel`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Retrieves the property IDs associated with a specific asset model.
     *
     * @param assetModelId the ID of the asset model that defines the properties.
     * @return a {@link CompletableFuture} that represents a {@link Map} result that
associates the property name to the
     *         propert ID. The calling code can attach callbacks, then handle the
result or exception by calling
     *         {@link CompletableFuture#join()} or {@link CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<Map<String, String>> getPropertyIds(String
assetModelId) {
```

```
        ListAssetModelPropertiesRequest modelPropertiesRequest =
ListAssetModelPropertiesRequest.builder().assetModelId(assetModelId).build();
        return getAsyncClient().listAssetModelProperties(modelPropertiesRequest)
            .handle((response, throwable) -> {
                if (response != null) {
                    return response.assetModelPropertySummaries().stream()
                        .collect(Collectors
                            .toMap(AssetModelPropertySummary::name,
AssetModelPropertySummary::id));
                } else {
                    logger.error("Error occurred while fetching property IDs: {}.",
throwable.getCause().getMessage());
                    throw (CompletionException) throwable;
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeAssetModel中的。

`DescribeGateway`

以下代码示例演示了如何使用 `DescribeGateway`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Describes the specified gateway.
     *
     * @param gatewayId the ID of the gateway to describe.
     * @return a {@link CompletableFuture} that represents a {@link
DescribeGatewayResponse} result. The calling code
     *         can attach callbacks, then handle the result or exception by calling
{@link CompletableFuture#join()} or
     *         {@link CompletableFuture#get()}.
```

```
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<DescribeGatewayResponse> describeGatewayAsync(String
gatewayId) {
        DescribeGatewayRequest request = DescribeGatewayRequest.builder()
            .gatewayId(gatewayId)
            .build();

        return getAsyncClient().describeGateway(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    logger.error("An error occurred during the describeGateway
method: {}", exception.getCause().getMessage());
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeGateway中的。

`GetAssetPropertyValue`

以下代码示例演示了如何使用 `GetAssetPropertyValue`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Fetches the value of an asset property.
     *
```

```
    * @param propId  the ID of the asset property to fetch.
    * @param assetId the ID of the asset to fetch the property value for.
    * @return a {@link CompletableFuture} that represents a {@link Double} result.
The calling code can attach
    *         callbacks, then handle the result or exception by calling {@link
CompletableFuture#join()} or
    *         {@link CompletableFuture#get()}.
    *         <p>
    *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
    *         it available to the calling code as a {@link CompletionException}. By
calling
    *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
    */
   public CompletableFuture<Double> getAssetPropValueAsync(String propId, String
assetId) {
       GetAssetPropertyValueRequest assetPropertyValueRequest =
GetAssetPropertyValueRequest.builder()
               .propertyId(propId)
               .assetId(assetId)
               .build();

       return getAsyncClient().getAssetPropertyValue(assetPropertyValueRequest)
               .handle((response, exception) -> {
                   if (exception != null) {
                       logger.error("Error occurred while fetching property value:
{}.", exception.getCause().getMessage());
                       throw (CompletionException) exception;
                   }
                   return response.propertyValue().value().doubleValue();
               });
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetAssetPropertyValue中的。

`ListAssetModels`

以下代码示例演示了如何使用 `ListAssetModels`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Retrieves the asset model ID for the given asset model name.
     *
     * @param assetModelName the name of the asset model for the ID.
     * @return a {@link CompletableFuture} that represents a {@link String} result
of the asset model ID or null if the
     *         asset model cannot be found. The calling code can attach callbacks,
then handle the result or exception
     *         by calling {@link CompletableFuture#join()} or {@link
CompletableFuture#get()}.
     *         <p>
     *         If any completion stage in this method throws an exception, the
method logs the exception cause and keeps
     *         it available to the calling code as a {@link CompletionException}. By
calling
     *         {@link CompletionException#getCause()}, the calling code can access
the original exception.
     */
    public CompletableFuture<String> getAssetModelIdAsync(String assetModelName) {
        ListAssetModelsRequest listAssetModelsRequest =
ListAssetModelsRequest.builder().build();
        return getAsyncClient().listAssetModels(listAssetModelsRequest)
                .handle((listAssetModelsResponse, exception) -> {
                    if (exception != null) {
                        logger.error("Failed to retrieve Asset Model ID: {}",
exception.getCause().getMessage());
                        throw (CompletionException) exception;
                    }
                    for (AssetModelSummary assetModelSummary :
listAssetModelsResponse.assetModelSummaries()) {
                        if (assetModelSummary.name().equals(assetModelName)) {
                            return assetModelSummary.id();
                        }
                    }
```

```
                    return null;
                });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListAssetModels中的。

使用 SDK for 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。

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 基本功能
- 操作

开始使用

开始使用 Amazon Keysp

以下代码示例展示了如何开始使用 Amazon Keyspaces。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListKeyspaces中的。

基本功能

了解基本功能

以下代码示例展示了如何：

- 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。添加一列来跟踪观看的电影。
- 将表还原到以前的状态并清理资源。

适用于 Java 的 SDK 2.x

> **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。

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * Before running this Java code example, you must create a
 * Java keystore (JKS) file and place it in your project's resources folder.
 *
 * This file is a secure file format used to hold certificate information for
```

```
 * Java applications. This is required to make a connection to Amazon Keyspaces.
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/keyspaces/latest/devguide/using_java_driver.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Create a keyspace.
 * 2. Check for keyspace existence.
 * 3. List keyspaces using a paginator.
 * 4. Create a table with a simple movie data schema and enable point-in-time
 * recovery.
 * 5. Check for the table to be in an Active state.
 * 6. List all tables in the keyspace.
 * 7. Use a Cassandra driver to insert some records into the Movie table.
 * 8. Get all records from the Movie table.
 * 9. Get a specific Movie.
 * 10. Get a UTC timestamp for the current time.
 * 11. Update the table schema to add a ‘watched’ Boolean column.
 * 12. Update an item as watched.
 * 13. Query for items with watched = True.
 * 14. Restore the table back to the previous state using the timestamp.
 * 15. Check for completion of the restore action.
 * 16. Delete the table.
 * 17. Confirm that both tables are deleted.
 * 18. Delete the keyspace.
 */

public class ScenarioKeyspaces {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    /*
     * Usage:
     * fileName - The name of the JSON file that contains movie data. (Get this file
     * from the GitHub repo at resources/sample_file.)
     * keyspaceName - The name of the keyspace to create.
     */
    public static void main(String[] args) throws InterruptedException, IOException
 {
        String fileName = "<Replace with the JSON file that contains movie data>";
        String keyspaceName = "<Replace with the name of the keyspace to create>";
        String titleUpdate = "The Family";
        int yearUpdate = 2013;
        String tableName = "Movie";
```

```
        String tableNameRestore = "MovieRestore";
        Region region = Region.US_EAST_1;
        KeyspacesClient keyClient = KeyspacesClient.builder()
                .region(region)
                .build();

        DriverConfigLoader loader =
DriverConfigLoader.fromClasspath("application.conf");
        CqlSession session = CqlSession.builder()
                .withConfigLoader(loader)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Keyspaces example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create a keyspace.");
        createKeySpace(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        Thread.sleep(5000);
        System.out.println("2. Check for keyspace existence.");
        checkKeyspaceExistence(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. List keyspaces using a paginator.");
        listKeyspacesPaginator(keyClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Create a table with a simple movie data schema and
enable point-in-time recovery.");
        createTable(keyClient, keyspaceName, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Check for the table to be in an Active state.");
        Thread.sleep(6000);
        checkTable(keyClient, keyspaceName, tableName);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("6. List all tables in the keyspace.");
        listTables(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Use a Cassandra driver to insert some records into
the Movie table.");
        Thread.sleep(6000);
        loadData(session, fileName, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Get all records from the Movie table.");
        getMovieData(session, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get a specific Movie.");
        getSpecificMovie(session, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get a UTC timestamp for the current time.");
        ZonedDateTime utc = ZonedDateTime.now(ZoneOffset.UTC);
        System.out.println("DATETIME = " + Date.from(utc.toInstant()));
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Update the table schema to add a watched Boolean
column.");
        updateTable(keyClient, keyspaceName, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Update an item as watched.");
        Thread.sleep(10000); // Wait 10 secs for the update.
        updateRecord(session, keyspaceName, titleUpdate, yearUpdate);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Query for items with watched = True.");
        getWatchedData(session, keyspaceName);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("14. Restore the table back to the previous state using
the timestamp.");
        System.out.println("Note that the restore operation can take up to 20
minutes.");
        restoreTable(keyClient, keyspaceName, utc);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Check for completion of the restore action.");
        Thread.sleep(5000);
        checkRestoredTable(keyClient, keyspaceName, "MovieRestore");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Delete both tables.");
        deleteTable(keyClient, keyspaceName, tableName);
        deleteTable(keyClient, keyspaceName, tableNameRestore);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Confirm that both tables are deleted.");
        checkTableDelete(keyClient, keyspaceName, tableName);
        checkTableDelete(keyClient, keyspaceName, tableNameRestore);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Delete the keyspace.");
        deleteKeyspace(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The scenario has completed successfully.");
        System.out.println(DASHES);
    }

    public static void deleteKeyspace(KeyspacesClient keyClient, String
keyspaceName) {
        try {
            DeleteKeyspaceRequest deleteKeyspaceRequest =
DeleteKeyspaceRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();
```

```
            keyClient.deleteKeyspace(deleteKeyspaceRequest);

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkTableDelete(KeyspacesClient keyClient, String
keyspaceName, String tableName)
            throws InterruptedException {
        try {
            String status;
            GetTableResponse response;
            GetTableRequest tableRequest = GetTableRequest.builder()
                    .keyspaceName(keyspaceName)
                    .tableName(tableName)
                    .build();

            // Keep looping until table cannot be found and a
ResourceNotFoundException is
            // thrown.
            while (true) {
                response = keyClient.getTable(tableRequest);
                status = response.statusAsString();
                System.out.println(". The table status is " + status);
                Thread.sleep(500);
            }

        } catch (ResourceNotFoundException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println("The table is deleted");
    }

    public static void deleteTable(KeyspacesClient keyClient, String keyspaceName,
String tableName) {
        try {
            DeleteTableRequest tableRequest = DeleteTableRequest.builder()
                    .keyspaceName(keyspaceName)
                    .tableName(tableName)
                    .build();
```

```
            keyClient.deleteTable(tableRequest);

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkRestoredTable(KeyspacesClient keyClient, String
keyspaceName, String tableName)
            throws InterruptedException {
        try {
            boolean tableStatus = false;
            String status;
            GetTableResponse response = null;
            GetTableRequest tableRequest = GetTableRequest.builder()
                    .keyspaceName(keyspaceName)
                    .tableName(tableName)
                    .build();

            while (!tableStatus) {
                response = keyClient.getTable(tableRequest);
                status = response.statusAsString();
                System.out.println("The table status is " + status);

                if (status.compareTo("ACTIVE") == 0) {
                    tableStatus = true;
                }
                Thread.sleep(500);
            }

            List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
            for (ColumnDefinition def : cols) {
                System.out.println("The column name is " + def.name());
                System.out.println("The column type is " + def.type());
            }

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
   public static void restoreTable(KeyspacesClient keyClient, String keyspaceName,
ZonedDateTime utc) {
       try {
           Instant myTime = utc.toInstant();
           RestoreTableRequest restoreTableRequest = RestoreTableRequest.builder()
                   .restoreTimestamp(myTime)
                   .sourceTableName("Movie")
                   .targetKeyspaceName(keyspaceName)
                   .targetTableName("MovieRestore")
                   .sourceKeyspaceName(keyspaceName)
                   .build();

           RestoreTableResponse response =
keyClient.restoreTable(restoreTableRequest);
           System.out.println("The ARN of the restored table is " +
response.restoredTableARN());

       } catch (KeyspacesException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   public static void getWatchedData(CqlSession session, String keyspaceName) {
       ResultSet resultSet = session
               .execute("SELECT * FROM \"" + keyspaceName + "\".\"Movie\" WHERE
watched = true ALLOW FILTERING;");
       resultSet.forEach(item -> {
           System.out.println("The Movie title is " + item.getString("title"));
           System.out.println("The Movie year is " + item.getInt("year"));
           System.out.println("The plot is " + item.getString("plot"));
       });
   }

   public static void updateRecord(CqlSession session, String keySpace, String
titleUpdate, int yearUpdate) {
       String sqlStatement = "UPDATE \"" + keySpace
               + "\".\"Movie\" SET watched=true WHERE title = :k0 AND year = :k1;";
       BatchStatementBuilder builder =
BatchStatement.builder(DefaultBatchType.UNLOGGED);
       builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM);
       PreparedStatement preparedStatement = session.prepare(sqlStatement);
       builder.addStatement(preparedStatement.boundStatementBuilder()
               .setString("k0", titleUpdate)
```

```
                .setInt("k1", yearUpdate)
                .build());

        BatchStatement batchStatement = builder.build();
        session.execute(batchStatement);
    }

    public static void updateTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
        try {
            ColumnDefinition def = ColumnDefinition.builder()
                    .name("watched")
                    .type("boolean")
                    .build();

            UpdateTableRequest tableRequest = UpdateTableRequest.builder()
                    .keyspaceName(keySpace)
                    .tableName(tableName)
                    .addColumns(def)
                    .build();

            keyClient.updateTable(tableRequest);

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getSpecificMovie(CqlSession session, String keyspaceName) {
        ResultSet resultSet = session.execute(
                "SELECT * FROM \"" + keyspaceName + "\".\"Movie\" WHERE title = 'The
Family' ALLOW FILTERING ;");
        resultSet.forEach(item -> {
            System.out.println("The Movie title is " + item.getString("title"));
            System.out.println("The Movie year is " + item.getInt("year"));
            System.out.println("The plot is " + item.getString("plot"));
        });
    }

    // Get records from the Movie table.
    public static void getMovieData(CqlSession session, String keyspaceName) {
        ResultSet resultSet = session.execute("SELECT * FROM \"" + keyspaceName +
"\".\"Movie\";");
```

```
        resultSet.forEach(item -> {
            System.out.println("The Movie title is " + item.getString("title"));
            System.out.println("The Movie year is " + item.getInt("year"));
            System.out.println("The plot is " + item.getString("plot"));
        });
    }

    // Load data into the table.
    public static void loadData(CqlSession session, String fileName, String
keySpace) throws IOException {
        String sqlStatement = "INSERT INTO \"" + keySpace + "\".\"Movie\" (title,
year, plot) values (:k0, :k1, :k2)";
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0;
        while (iter.hasNext()) {

            // Add 20 movies to the table.
            if (t == 20)
                break;
            currentNode = (ObjectNode) iter.next();

            int year = currentNode.path("year").asInt();
            String title = currentNode.path("title").asText();
            String plot = currentNode.path("info").path("plot").toString();

            // Insert the data into the Amazon Keyspaces table.
            BatchStatementBuilder builder =
BatchStatement.builder(DefaultBatchType.UNLOGGED);
            builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM);
            PreparedStatement preparedStatement = session.prepare(sqlStatement);
            builder.addStatement(preparedStatement.boundStatementBuilder()
                    .setString("k0", title)
                    .setInt("k1", year)
                    .setString("k2", plot)
                    .build());

            BatchStatement batchStatement = builder.build();
            session.execute(batchStatement);
            t++;
        }
```

```
        System.out.println("You have added " + t + " records successfully!");
    }

    public static void listTables(KeyspacesClient keyClient, String keyspaceName) {
        try {
            ListTablesRequest tablesRequest = ListTablesRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            ListTablesIterable listRes =
keyClient.listTablesPaginator(tablesRequest);
            listRes.stream()
                    .flatMap(r -> r.tables().stream())
                    .forEach(content -> System.out.println(" ARN: " +
content.resourceArn() +
                            " Table name: " + content.tableName()));

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkTable(KeyspacesClient keyClient, String keyspaceName,
String tableName)
            throws InterruptedException {
        try {
            boolean tableStatus = false;
            String status;
            GetTableResponse response = null;
            GetTableRequest tableRequest = GetTableRequest.builder()
                    .keyspaceName(keyspaceName)
                    .tableName(tableName)
                    .build();

            while (!tableStatus) {
                response = keyClient.getTable(tableRequest);
                status = response.statusAsString();
                System.out.println(". The table status is " + status);

                if (status.compareTo("ACTIVE") == 0) {
                    tableStatus = true;
                }
```

```
                Thread.sleep(500);
            }

            List<ColumnDefinition> cols = response.schemaDefinition().allColumns();
            for (ColumnDefinition def : cols) {
                System.out.println("The column name is " + def.name());
                System.out.println("The column type is " + def.type());
            }

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
        try {
            // Set the columns.
            ColumnDefinition defTitle = ColumnDefinition.builder()
                    .name("title")
                    .type("text")
                    .build();

            ColumnDefinition defYear = ColumnDefinition.builder()
                    .name("year")
                    .type("int")
                    .build();

            ColumnDefinition defReleaseDate = ColumnDefinition.builder()
                    .name("release_date")
                    .type("timestamp")
                    .build();

            ColumnDefinition defPlot = ColumnDefinition.builder()
                    .name("plot")
                    .type("text")
                    .build();

            List<ColumnDefinition> colList = new ArrayList<>();
            colList.add(defTitle);
            colList.add(defYear);
            colList.add(defReleaseDate);
            colList.add(defPlot);
```

```
            // Set the keys.
            PartitionKey yearKey = PartitionKey.builder()
                    .name("year")
                    .build();

            PartitionKey titleKey = PartitionKey.builder()
                    .name("title")
                    .build();

            List<PartitionKey> keyList = new ArrayList<>();
            keyList.add(yearKey);
            keyList.add(titleKey);

            SchemaDefinition schemaDefinition = SchemaDefinition.builder()
                    .partitionKeys(keyList)
                    .allColumns(colList)
                    .build();

            PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
                    .status(PointInTimeRecoveryStatus.ENABLED)
                    .build();

            CreateTableRequest tableRequest = CreateTableRequest.builder()
                    .keyspaceName(keySpace)
                    .tableName(tableName)
                    .schemaDefinition(schemaDefinition)
                    .pointInTimeRecovery(timeRecovery)
                    .build();

            CreateTableResponse response = keyClient.createTable(tableRequest);
            System.out.println("The table ARN is " + response.resourceArn());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void listKeyspacesPaginator(KeyspacesClient keyClient) {
        try {
            ListKeyspacesRequest keyspacesRequest = ListKeyspacesRequest.builder()
                    .maxResults(10)
                    .build();
```

```
            ListKeyspacesIterable listRes =
keyClient.listKeyspacesPaginator(keyspacesRequest);
            listRes.stream()
                    .flatMap(r -> r.keyspaces().stream())
                    .forEach(content -> System.out.println(" Name: " +
content.keyspaceName()));

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkKeyspaceExistence(KeyspacesClient keyClient, String
keyspaceName) {
        try {
            GetKeyspaceRequest keyspaceRequest = GetKeyspaceRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            GetKeyspaceResponse response = keyClient.getKeyspace(keyspaceRequest);
            String name = response.keyspaceName();
            System.out.println("The " + name + " KeySpace is ready");

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createKeySpace(KeyspacesClient keyClient, String
keyspaceName) {
        try {
            CreateKeyspaceRequest keyspaceRequest = CreateKeyspaceRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            CreateKeyspaceResponse response =
keyClient.createKeyspace(keyspaceRequest);
            System.out.println("The ARN of the KeySpace is " +
response.resourceArn());

        } catch (KeyspacesException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - CreateKeyspace
  - CreateTable
  - DeleteKeyspace
  - DeleteTable
  - GetKeyspace
  - GetTable
  - ListKeyspaces
  - ListTables
  - RestoreTable
  - UpdateTable

操作

`CreateKeyspace`

以下代码示例演示了如何使用 `CreateKeyspace`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateKeyspace中的。

`CreateTable`

以下代码示例演示了如何使用 `CreateTable`。

适用于 Java 的 SDK 2.x

> **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。

```
    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);
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateTable中的。

`DeleteKeyspace`

以下代码示例演示了如何使用 `DeleteKeyspace`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteKeyspace中的。

`DeleteTable`

以下代码示例演示了如何使用 `DeleteTable`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteTable中的。

`GetKeyspace`

以下代码示例演示了如何使用 `GetKeyspace`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetKeyspace中的。

`GetTable`

以下代码示例演示了如何使用 `GetTable`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetTable中的。

ListKeyspaces

以下代码示例演示了如何使用 ListKeyspaces。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListKeyspaces中的。

`ListTables`

以下代码示例演示了如何使用 `ListTables`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListTables中的。

`RestoreTable`

以下代码示例演示了如何使用 `RestoreTable`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考RestoreTable中的。

`UpdateTable`

以下代码示例演示了如何使用 `UpdateTable`。

适用于 Java 的 SDK 2.x

> **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。

```
    public static void updateTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
        try {
            ColumnDefinition def = ColumnDefinition.builder()
                    .name("watched")
                    .type("boolean")
                    .build();

            UpdateTableRequest tableRequest = UpdateTableRequest.builder()
```

```
                    .keyspaceName(keySpace)
                    .tableName(tableName)
                    .addColumns(def)
                    .build();

            keyClient.updateTable(tableRequest);

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考UpdateTable中的。

使用 SDK for 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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从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`

以下代码示例演示了如何使用 `CreateStream`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateStream中的。

`DeleteStream`

以下代码示例演示了如何使用 `DeleteStream`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteStream中的。

`GetRecords`

以下代码示例演示了如何使用 `GetRecords`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetRecords中的。

PutRecord

以下代码示例演示了如何使用 PutRecord。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，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。

适用于 Java 的 SDK 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。

适用于 Java 的 SDK 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 使用适用于 Java 的 SDK 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

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 基本功能
- 操作

开始使用

你好 AWS KMS

以下代码示例展示了如何开始使用 AWS Key Management Service。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.services.kms.KmsAsyncClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.paginators.ListKeysPublisher;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloKMS {
    public static void main(String[] args) {
        listAllKeys();
    }

    public static void listAllKeys() {
        KmsAsyncClient kmsAsyncClient = KmsAsyncClient.builder()
            .build();
        ListKeysRequest listKeysRequest = ListKeysRequest.builder()
            .limit(15)
            .build();

        /*
         * The `subscribe` method is required when using paginator methods in the
 AWS SDK
         * because paginator methods return an instance of a `ListKeysPublisher`,
 which is
         * based on a reactive stream. This allows asynchronous retrieval of
 paginated
         * results as they become available. By subscribing to the stream, we can
 process
         * each page of results as they are emitted.
         */
        ListKeysPublisher keysPublisher =
 kmsAsyncClient.listKeysPaginator(listKeysRequest);
        CompletableFuture<Void> future = keysPublisher
            .subscribe(r -> r.keys().forEach(key ->
                System.out.println("The key ARN is: " + key.keyArn() + ". The key Id
 is: " + key.keyId())))
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    System.err.println("Error occurred: " + exception.getMessage());
                } else {
                    System.out.println("Successfully listed all keys.");
                }
            });

        try {
            future.join();
```

```
        } catch (Exception e) {
            System.err.println("Failed to list keys: " + e.getMessage());
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListKeys中的。

基本功能

了解基本功能

以下代码示例展示了如何：

- 创建 KMS 密钥。
- 列出您账户的 KMS 密钥并获取有关它们的详细信息。
- 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。
- 标签键。
- 删除 KMS 密钥。

适用于 Java 的 SDK 2.x

> **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场景。

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.kms.model.AlreadyExistsException;
```

```
import software.amazon.awssdk.services.kms.model.DisabledException;
import software.amazon.awssdk.services.kms.model.EnableKeyRotationResponse;
import software.amazon.awssdk.services.kms.model.KmsException;
import software.amazon.awssdk.services.kms.model.NotFoundException;
import software.amazon.awssdk.services.kms.model.RevokeGrantResponse;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class KMSScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static String accountId = "";

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

    static KMSActions kmsActions = new KMSActions();

    static Scanner scanner = new Scanner(System.in);

    static String aliasName = "alias/dev-encryption-key";

    public static void main(String[] args) {
        final String usage = """
            Usage: <granteePrincipal>

            Where:
               granteePrincipal - The principal (user, service account, or group) to
 whom the grant or permission is being given.
            """;

        if (args.length != 1) {
            logger.info(usage);
            return;
        }
```

```
        String granteePrincipal = args[0];
        String policyName = "default";

        accountId = kmsActions.getAccountId();
        String keyDesc = "Created by the AWS KMS API";

        logger.info(DASHES);
        logger.info("""
            Welcome to the AWS Key Management SDK Basics scenario.

            This program demonstrates how to interact with AWS Key Management using
the AWS SDK for Java (v2).
            The AWS Key Management Service (KMS) is a secure and highly available
service that allows you to create
            and manage AWS KMS keys and control their use across a wide range of AWS
services and applications.
            KMS provides a centralized and unified approach to managing encryption
keys, making it easier to meet your
            data protection and regulatory compliance requirements.

            This Basics scenario creates two key types:

            - A symmetric encryption key is used to encrypt and decrypt data.
            - An asymmetric key used to digitally sign data.

            Let's get started...
            """);
        waitForInputToContinue(scanner);

        try {
        // Run the methods that belong to this scenario.
        String targetKeyId = runScenario(granteePrincipal, keyDesc, policyName);
        requestDeleteResources(aliasName, targetKeyId);

        } catch (Throwable rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }
```

```
   private static String runScenario(String granteePrincipal, String keyDesc,
String policyName) throws Throwable {
       logger.info(DASHES);
       logger.info("1. Create a symmetric KMS key\n");
       logger.info("First, the program will creates a symmetric KMS key that you
can used to encrypt and decrypt data.");
       waitForInputToContinue(scanner);
       String targetKeyId;
       try {
           CompletableFuture<String> futureKeyId =
kmsActions.createKeyAsync(keyDesc);
           targetKeyId = futureKeyId.join();
           logger.info("A symmetric key was successfully created " + targetKeyId);

       } catch (RuntimeException rt) {
           Throwable cause = rt.getCause();
           if (cause instanceof KmsException kmsEx) {
               logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
           } else {
               logger.info("An unexpected error occurred: " + rt.getMessage());
           }
           throw cause;
       }
       waitForInputToContinue(scanner);

       logger.info(DASHES);
       logger.info("""
           2. Enable a KMS key

           By default, when the SDK creates an AWS key, it is enabled. The next bit
of code checks to
           determine if the key is enabled.
            """);
       waitForInputToContinue(scanner);
       boolean isEnabled;
       try {
           CompletableFuture<Boolean> futureIsKeyEnabled =
kmsActions.isKeyEnabledAsync(targetKeyId);
           isEnabled = futureIsKeyEnabled.join();
           logger.info("Is the key enabled? {}", isEnabled);

       } catch (RuntimeException rt) {
```

```
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }

        if (!isEnabled)
            try {
                CompletableFuture<Void> future =
kmsActions.enableKeyAsync(targetKeyId);
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: " + rt.getMessage());
                }
                throw cause;
            }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("3. Encrypt data using the symmetric KMS key");
        String plaintext = "Hello, AWS KMS!";
        logger.info("""
            One of the main uses of symmetric keys is to encrypt and decrypt data.
            Next, the code encrypts the string {} with the SYMMETRIC_DEFAULT
encryption algorithm.
            """, plaintext);
        waitForInputToContinue(scanner);
        SdkBytes encryptedData;
        try {
            CompletableFuture<SdkBytes> future =
kmsActions.encryptDataAsync(targetKeyId, plaintext);
            encryptedData = future.join();

        } catch (RuntimeException rt) {
```

```
            Throwable cause = rt.getCause();
            if (cause instanceof DisabledException kmsDisabledEx) {
                logger.info("KMS error occurred due to a disabled
key: Error message: {}, Error code {}", kmsDisabledEx.getMessage(),
kmsDisabledEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("4. Create an alias");
        logger.info("""

            The alias name should be prefixed with 'alias/'.
            The default, 'alias/dev-encryption-key'.
             """);
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<Void> future =
kmsActions.createCustomAliasAsync(targetKeyId, aliasName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof AlreadyExistsException kmsExistsEx) {
                if (kmsExistsEx.getMessage().contains("already exists")) {
                    logger.info("The alias '" + aliasName + "' already exists.
Moving on...");
                }
            } else {
                logger.error("An unexpected error occurred: " + rt.getMessage(),
rt);
                deleteKey(targetKeyId);
                throw cause;
            }
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
```

```
        logger.info("5. List all of your aliases");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Object> future = kmsActions.listAllAliasesAsync();
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("6. Enable automatic rotation of the KMS key");
        logger.info("""

            By default, when the SDK enables automatic rotation of a KMS key,
            KMS rotates the key material of the KMS key one year (approximately 365
days) from the enable date and every year
            thereafter.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<EnableKeyRotationResponse> future =
kmsActions.enableKeyRotationAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
```

```
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("""
            7. Create a grant

            A grant is a policy instrument that allows Amazon Web Services
principals to use KMS keys.
            It also can allow them to view a KMS key (DescribeKey) and create and
manage grants.
            When authorizing access to a KMS key, grants are considered along with
key policies and IAM policies.
            """);

        waitForInputToContinue(scanner);
        String grantId = null;
        try {
            CompletableFuture<String> futureGrantId =
kmsActions.grantKeyAsync(targetKeyId, granteePrincipal);
            grantId = futureGrantId.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. List grants for the KMS key");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Object> future =
kmsActions.displayGrantIdsAsync(targetKeyId);
```

```
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("9. Revoke the grant");
        logger.info("""
            The revocation of a grant immediately removes the permissions and access
that the grant had provided.
            This means that any principal (user, role, or service) that was granted
access to perform specific
            KMS operations on a KMS key will no longer be able to perform those
operations.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<RevokeGrantResponse> future =
kmsActions.revokeKeyGrantAsync(targetKeyId, grantId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                if (kmsEx.getMessage().contains("Grant does not exist")) {
                    logger.info("The grant ID '" + grantId + "' does not exist.
Moving on...");
                } else {
                    logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                    throw cause;
                }
            } else {
```

```
                logger.info("An unexpected error occurred: " + rt.getMessage());
                deleteAliasName(aliasName);
                deleteKey(targetKeyId);
                throw cause;
            }
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("10. Decrypt the data\n");
        logger.info("""
            Lets decrypt the data that was encrypted in an early step.
            The code uses the same key to decrypt the string that we encrypted
earlier in the program.
            """);
        waitForInputToContinue(scanner);
        String decryptedData = "";
        try {
            CompletableFuture<String> future =
kmsActions.decryptDataAsync(encryptedData, targetKeyId);
            decryptedData = future.join();
            logger.info("Decrypted data: " + decryptedData);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        logger.info("Decrypted text is: " + decryptedData);
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("11. Replace a key policy\n");
        logger.info("""
            A key policy is a resource policy for a KMS key. Key policies are the
primary way to control
```

```
            access to KMS keys. Every KMS key must have exactly one key policy. The
statements in the key policy
            determine who has permission to use the KMS key and how they can use
it.
            You can also use IAM policies and grants to control access to the KMS
key, but every KMS key
            must have a key policy.

            By default, when you create a key by using the SDK, a policy is created
that
            gives the AWS account that owns the KMS key full access to the KMS key.

            Let's try to replace the automatically created policy with the following
policy.

                "Version":"2012-10-17",
                "Statement": [{
                "Effect": "Allow",
                "Principal": {"AWS": "arn:aws:iam::0000000000:root"},
                "Action": "kms:*",
                "Resource": "*"
                }]
            """);

        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Boolean> future =
kmsActions.replacePolicyAsync(targetKeyId, policyName, accountId);
            boolean success = future.join();
            if (success) {
                logger.info("Key policy replacement succeeded.");
            } else {
                logger.error("Key policy replacement failed.");
            }

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
```

```
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("12. Get the key policy\n");
        logger.info("The next bit of code that runs gets the key policy to make sure
it exists.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
kmsActions.getKeyPolicyAsync(targetKeyId, policyName);
            String policy = future.join();
            if (!policy.isEmpty()) {
                logger.info("Retrieved policy: " + policy);
            }

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("13. Create an asymmetric KMS key and sign your data\n");
        logger.info("""
             Signing your data with an AWS key can provide several benefits that
make it an attractive option
             for your data signing needs. By using an AWS KMS key, you can leverage
the
             security controls and compliance features provided by AWS,
             which can help you meet various regulatory requirements and enhance the
overall security posture
             of your organization.
            """);
```

```
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Boolean> future = kmsActions.signVerifyDataAsync();
            boolean success = future.join();
            if (success) {
                logger.info("Sign and verify data operation succeeded.");
            } else {
                logger.error("Sign and verify data operation failed.");
            }

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);

        logger.info(DASHES);
        logger.info("14. Tag your symmetric KMS Key\n");
        logger.info("""
            By using tags, you can improve the overall management, security, and
governance of your
            KMS keys, making it easier to organize, track, and control access to
your encrypted data within
            your AWS environment
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future = kmsActions.tagKMSKeyAsync(targetKeyId);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof KmsException kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
            } else {
```

```
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            deleteAliasName(aliasName);
            deleteKey(targetKeyId);
            throw cause;
        }
        waitForInputToContinue(scanner);
        return targetKeyId;
    }

    // Deletes KMS resources with user input.
    private static void requestDeleteResources(String aliasName, String targetKeyId)
{
        logger.info(DASHES);
        logger.info("15. Schedule the deletion of the KMS key\n");
        logger.info("""
            By default, KMS applies a waiting period of 30 days,
            but you can specify a waiting period of 7-30 days. When this operation
is successful,
            the key state of the KMS key changes to PendingDeletion and the key
can't be used in any
            cryptographic operations. It remains in this state for the duration of
the waiting period.

            Deleting a KMS key is a destructive and potentially dangerous operation.
When a KMS key is deleted,
            all data that was encrypted under the KMS key is unrecoverable.
            """);
        logger.info("Would you like to delete the Key Management resources? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete the AWS KMS resources.");
            waitForInputToContinue(scanner);
            try {
                CompletableFuture<Void> future =
kmsActions.deleteSpecificAliasAsync(aliasName);
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
```

```
                    logger.info("An unexpected error occurred: " + rt.getMessage());
                }
            }
            waitForInputToContinue(scanner);
            try {
                CompletableFuture<Void> future =
kmsActions.disableKeyAsync(targetKeyId);
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: " + rt.getMessage());
                }
            }

            try {
                CompletableFuture<Void> future =
kmsActions.deleteKeyAsync(targetKeyId);
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof KmsException kmsEx) {
                    logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: " + rt.getMessage());
                }
            }

        } else {
            logger.info("The Key Management resources will not be deleted");
        }

        logger.info(DASHES);
        logger.info("This concludes the AWS Key Management SDK scenario");
        logger.info(DASHES);
    }

    // This method is invoked from Exceptions to clean up the resources.
```

```
   private static void deleteKey(String targetKeyId) {
       try {
           CompletableFuture<Void> future =
kmsActions.disableKeyAsync(targetKeyId);
           future.join();

       } catch (RuntimeException rt) {
           Throwable cause = rt.getCause();
           if (cause instanceof KmsException kmsEx) {
               logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
           } else {
               logger.info("An unexpected error occurred: " + rt.getMessage());
           }
       }

       try {
           CompletableFuture<Void> future = kmsActions.deleteKeyAsync(targetKeyId);
           future.join();

       } catch (RuntimeException rt) {
           Throwable cause = rt.getCause();
           if (cause instanceof KmsException kmsEx) {
               logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
           } else {
               logger.info("An unexpected error occurred: " + rt.getMessage());
           }
       }
   }

   // This method is invoked from Exceptions to clean up the resources.
   private static void deleteAliasName(String aliasName) {
       try {
           CompletableFuture<Void> future =
kmsActions.deleteSpecificAliasAsync(aliasName);
           future.join();

       } catch (RuntimeException rt) {
           Throwable cause = rt.getCause();
           if (cause instanceof KmsException kmsEx) {
               logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
           } else {
```

```
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

定义一个包装 KMS 操作的类。

```
public class KMSActions {
    private static final Logger logger = LoggerFactory.getLogger(KMSActions.class);
    private static KmsAsyncClient kmsAsyncClient;

    /**
     * Retrieves an asynchronous AWS Key Management Service (KMS) client.
     * <p>
     * This method creates and returns a singleton instance of the KMS async client,
 with the following configurations:
     * <ul>
     *   <li>Max concurrency: 100</li>
     *   <li>Connection timeout: 60 seconds</li>
     *   <li>Read timeout: 60 seconds</li>
     *   <li>Write timeout: 60 seconds</li>
     *   <li>API call timeout: 2 minutes</li>
     *   <li>API call attempt timeout: 90 seconds</li>
     *   <li>Retry policy: up to 3 retries</li>
```

```
     *   <li>Credentials provider: environment variable credentials provider</li>
     * </ul>
     * <p>
     * If the client instance has already been created, it is returned instead of
creating a new one.
     *
     * @return the KMS async client instance
     */
    private static KmsAsyncClient getAsyncClient() {
        if (kmsAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryPolicy(RetryPolicy.builder()
                    .numRetries(3)
                    .build())
                .build();

            kmsAsyncClient = KmsAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return kmsAsyncClient;
    }

    /**
     * Creates a new symmetric encryption key asynchronously.
     *
     * @param keyDesc the description of the key to be created
     * @return a {@link CompletableFuture} that completes with the ID of the newly
created key
     * @throws RuntimeException if an error occurs while creating the key
     */
    public CompletableFuture<String> createKeyAsync(String keyDesc) {
        CreateKeyRequest keyRequest = CreateKeyRequest.builder()
```

```
            .description(keyDesc)
            .keySpec(KeySpec.SYMMETRIC_DEFAULT)
            .keyUsage(KeyUsageType.ENCRYPT_DECRYPT)
            .build();

        return getAsyncClient().createKey(keyRequest)
            .thenApply(resp -> resp.keyMetadata().keyId())
            .exceptionally(ex -> {
                throw new RuntimeException("An error occurred while creating the
key: " + ex.getMessage(), ex);
            });
    }

    /**
     * Asynchronously checks if a specified key is enabled.
     *
     * @param keyId the ID of the key to check
     * @return a {@link CompletableFuture} that, when completed, indicates whether
the key is enabled or not
     *
     * @throws RuntimeException if an exception occurs while checking the key state
     */
    public CompletableFuture<Boolean> isKeyEnabledAsync(String keyId) {
        DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
            .keyId(keyId)
            .build();

        CompletableFuture<DescribeKeyResponse> responseFuture =
getAsyncClient().describeKey(keyRequest);
        return responseFuture.whenComplete((resp, ex) -> {
            if (resp != null) {
                KeyState keyState = resp.keyMetadata().keyState();
                if (keyState == KeyState.ENABLED) {
                    logger.info("The key is enabled.");
                } else {
                    logger.info("The key is not enabled. Key state: {}", keyState);
                }
            } else {
                throw new RuntimeException(ex);
            }
        }).thenApply(resp -> resp.keyMetadata().keyState() == KeyState.ENABLED);
    }

    /**
```

```
    * Asynchronously enables the specified key.
    *
    * @param keyId the ID of the key to enable
    * @return a {@link CompletableFuture} that completes when the key has been
enabled
    */
   public CompletableFuture<Void> enableKeyAsync(String keyId) {
       EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
           .keyId(keyId)
           .build();

       CompletableFuture<EnableKeyResponse> responseFuture =
getAsyncClient().enableKey(enableKeyRequest);
       responseFuture.whenComplete((response, exception) -> {
           if (exception == null) {
               logger.info("Key with ID [{}] has been enabled.", keyId);
           } else {
               if (exception instanceof KmsException kmsEx) {
                   throw new RuntimeException("KMS error occurred while enabling
key: " + kmsEx.getMessage(), kmsEx);
               } else {
                   throw new RuntimeException("An unexpected error occurred while
enabling key: " + exception.getMessage(), exception);
               }
           }
       });

       return responseFuture.thenApply(response -> null);
   }

   /**
    * Encrypts the given text asynchronously using the specified KMS client and key
ID.
    *
    * @param keyId the ID of the KMS key to use for encryption
    * @param text the text to encrypt
    * @return a CompletableFuture that completes with the encrypted data as an
SdkBytes object
    */
   public CompletableFuture<SdkBytes> encryptDataAsync(String keyId, String text) {
       SdkBytes myBytes = SdkBytes.fromUtf8String(text);
       EncryptRequest encryptRequest = EncryptRequest.builder()
           .keyId(keyId)
           .plaintext(myBytes)
```

```
            .build();

        CompletableFuture<EncryptResponse> responseFuture =
getAsyncClient().encrypt(encryptRequest).toCompletableFuture();
        return responseFuture.whenComplete((response, ex) -> {
            if (response != null) {
                String algorithm = response.encryptionAlgorithm().toString();
                logger.info("The string was encrypted with algorithm {}.",
algorithm);
            } else {
                throw new RuntimeException(ex);
            }
        }).thenApply(EncryptResponse::ciphertextBlob);
    }

    /**
     * Creates a custom alias for the specified target key asynchronously.
     *
     * @param targetKeyId the ID of the target key for the alias
     * @param aliasName   the name of the alias to create
     * @return a {@link CompletableFuture} that completes when the alias creation
operation is finished
     */
    public CompletableFuture<Void> createCustomAliasAsync(String targetKeyId, String
aliasName) {
        CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
            .aliasName(aliasName)
            .targetKeyId(targetKeyId)
            .build();

        CompletableFuture<CreateAliasResponse> responseFuture =
getAsyncClient().createAlias(aliasRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("{} was successfully created.", aliasName);
            } else {
                if (exception instanceof ResourceExistsException) {
                    logger.info("Alias [{}] already exists. Moving on...",
aliasName);
                } else if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while creating
alias: " + kmsEx.getMessage(), kmsEx);
                } else {
```

```
                    throw new RuntimeException("An unexpected error occurred while
creating alias: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(response -> null);
    }

    /**
     * Asynchronously lists all the aliases in the current AWS account.
     *
     * @return a {@link CompletableFuture} that completes when the list of aliases
has been processed
     */
    public CompletableFuture<Object> listAllAliasesAsync() {
        ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
            .limit(15)
            .build();

        ListAliasesPublisher paginator =
getAsyncClient().listAliasesPaginator(aliasesRequest);
        return paginator.subscribe(response -> {
                response.aliases().forEach(alias ->
                    logger.info("The alias name is: " + alias.aliasName())
                );
            })
            .thenApply(v -> null)
            .exceptionally(ex -> {
                if (ex.getCause() instanceof KmsException) {
                    KmsException e = (KmsException) ex.getCause();
                    throw new RuntimeException("A KMS exception occurred: " +
e.getMessage());
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage());
                }
            });
    }

    /**
     * Enables key rotation asynchronously for the specified key ID.
     *
     * @param keyId the ID of the key for which to enable key rotation
```

```
    * @return a CompletableFuture that represents the asynchronous operation of
enabling key rotation
    * @throws RuntimeException if there was an error enabling key rotation, either
due to a KMS exception or an unexpected error
    */
   public CompletableFuture<EnableKeyRotationResponse>
enableKeyRotationAsync(String keyId) {
       EnableKeyRotationRequest enableKeyRotationRequest =
EnableKeyRotationRequest.builder()
           .keyId(keyId)
           .build();

       CompletableFuture<EnableKeyRotationResponse> responseFuture =
getAsyncClient().enableKeyRotation(enableKeyRotationRequest);
       responseFuture.whenComplete((response, exception) -> {
           if (exception == null) {
               logger.info("Key rotation has been enabled for key with id [{}]",
keyId);
           } else {
               if (exception instanceof KmsException kmsEx) {
                   throw new RuntimeException("Failed to enable key rotation: " +
kmsEx.getMessage(), kmsEx);
               } else {
                   throw new RuntimeException("An unexpected error occurred: " +
exception.getMessage(), exception);
               }
           }
       });

       return responseFuture;
   }

   /**
    * Grants permissions to a specified principal on a customer master key (CMK)
asynchronously.
    *
    * @param keyId             The unique identifier for the customer master key
(CMK) that the grant applies to.
    * @param granteePrincipal  The principal that is given permission to perform
the operations that the grant permits on the CMK.
    * @return A {@link CompletableFuture} that, when completed, contains the ID of
the created grant.
    * @throws RuntimeException If an error occurs during the grant creation
process.
```

```
     */
    public CompletableFuture<String> grantKeyAsync(String keyId, String
granteePrincipal) {
        List<GrantOperation> grantPermissions = List.of(
            GrantOperation.ENCRYPT,
            GrantOperation.DECRYPT,
            GrantOperation.DESCRIBE_KEY
        );

        CreateGrantRequest grantRequest = CreateGrantRequest.builder()
            .keyId(keyId)
            .name("grant1")
            .granteePrincipal(granteePrincipal)
            .operations(grantPermissions)
            .build();

        CompletableFuture<CreateGrantResponse> responseFuture =
getAsyncClient().createGrant(grantRequest);
        responseFuture.whenComplete((response, ex) -> {
            if (ex == null) {
                logger.info("Grant created successfully with ID: " +
response.grantId());
            } else {
                if (ex instanceof KmsException kmsEx) {
                    throw new RuntimeException("Failed to create grant: " +
kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage(), ex);
                }
            }
        });

        return responseFuture.thenApply(CreateGrantResponse::grantId);
    }

    /**
     * Asynchronously displays the grant IDs for the specified key ID.
     *
     * @param keyId the ID of the AWS KMS key for which to list the grants
     * @return a {@link CompletableFuture} that, when completed, will be null if
the operation succeeded, or will throw a {@link RuntimeException} if the operation
failed
```

```
    * @throws RuntimeException if there was an error listing the grants, either due
to an {@link KmsException} or an unexpected error
    */
   public CompletableFuture<Object> displayGrantIdsAsync(String keyId) {
       ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
           .keyId(keyId)
           .limit(15)
           .build();

       ListGrantsPublisher paginator =
getAsyncClient().listGrantsPaginator(grantsRequest);
       return paginator.subscribe(response -> {
               response.grants().forEach(grant -> {
                   logger.info("The grant Id is: " + grant.grantId());
               });
           })
           .thenApply(v -> null)
           .exceptionally(ex -> {
               Throwable cause = ex.getCause();
               if (cause instanceof KmsException) {
                   throw new RuntimeException("Failed to list grants: " +
cause.getMessage(), cause);
               } else {
                   throw new RuntimeException("An unexpected error occurred: " +
cause.getMessage(), cause);
               }
           });
   }

   /**
    * Revokes a grant for the specified AWS KMS key asynchronously.
    *
    * @param keyId   The ID or key ARN of the AWS KMS key.
    * @param grantId The identifier of the grant to be revoked.
    * @return A {@link CompletableFuture} representing the asynchronous operation
of revoking the grant.
    *         The {@link CompletableFuture} will complete with a {@link
RevokeGrantResponse} object
    *         if the operation is successful, or with a {@code null} value if an
error occurs.
    */
   public CompletableFuture<RevokeGrantResponse> revokeKeyGrantAsync(String keyId,
String grantId) {
       RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
```

```
            .keyId(keyId)
            .grantId(grantId)
            .build();

        CompletableFuture<RevokeGrantResponse> responseFuture =
getAsyncClient().revokeGrant(grantRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("Grant ID: [" + grantId + "] was successfully
revoked!");
            } else {
                if (exception instanceof KmsException kmsEx) {
                    if (kmsEx.getMessage().contains("Grant does not exist")) {
                        logger.info("The grant ID '" + grantId + "' does not exist.
Moving on...");
                    } else {
                        throw new RuntimeException("KMS error occurred: " +
kmsEx.getMessage(), kmsEx);
                    }
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
exception.getMessage(), exception);
                }
            }
        });

        return responseFuture;
    }


    /**
     * Asynchronously decrypts the given encrypted data using the specified key ID.
     *
     * @param encryptedData The encrypted data to be decrypted.
     * @param keyId The ID of the key to be used for decryption.
     * @return A CompletableFuture that, when completed, will contain the decrypted
data as a String.
     *         If an error occurs during the decryption process, the
CompletableFuture will complete
     *         exceptionally with the error, and the method will return an empty
String.
     */
    public CompletableFuture<String> decryptDataAsync(SdkBytes encryptedData, String
keyId) {
```

```
        DecryptRequest decryptRequest = DecryptRequest.builder()
            .ciphertextBlob(encryptedData)
            .keyId(keyId)
            .build();

        CompletableFuture<DecryptResponse> responseFuture =
getAsyncClient().decrypt(decryptRequest);
        responseFuture.whenComplete((decryptResponse, exception) -> {
            if (exception == null) {
                logger.info("Data decrypted successfully for key ID: " + keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while decrypting
data: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred while
decrypting data: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(decryptResponse ->
decryptResponse.plaintext().asString(StandardCharsets.UTF_8));
    }

    /**
     * Asynchronously replaces the policy for the specified KMS key.
     *
     * @param keyId       the ID of the KMS key to replace the policy for
     * @param policyName  the name of the policy to be replaced
     * @param accountId   the AWS account ID to be used in the policy
     * @return a {@link CompletableFuture} that completes with a boolean indicating
     *         whether the policy replacement was successful or not
     */
    public CompletableFuture<Boolean> replacePolicyAsync(String keyId, String
policyName, String accountId) {
        String policy = """
    {
      "Version":"2012-10-17",
      "Statement": [{
        "Effect": "Allow",
        "Principal": {"AWS": "arn:aws:iam::%s:root"},
        "Action": "kms:*",
        "Resource": "*"
```

```
      }]
    }
    """.formatted(accountId);

        PutKeyPolicyRequest keyPolicyRequest = PutKeyPolicyRequest.builder()
            .keyId(keyId)
            .policyName(policyName)
            .policy(policy)
            .build();

        // First, get the current policy to check if it exists
        return getAsyncClient().getKeyPolicy(r ->
r.keyId(keyId).policyName(policyName))
            .thenCompose(response -> {
                logger.info("Current policy exists. Replacing it...");
                return getAsyncClient().putKeyPolicy(keyPolicyRequest);
            })
            .thenApply(putPolicyResponse -> {
                logger.info("The key policy has been replaced.");
                return true;
            })
            .exceptionally(throwable -> {
                if (throwable.getCause() instanceof LimitExceededException) {
                    logger.error("Cannot replace policy, as only one policy is
allowed per key.");
                    return false;
                }
                throw new RuntimeException("Error replacing policy", throwable);
            });
    }


    /**
     * Asynchronously retrieves the key policy for the specified key ID and policy
name.
     *
     * @param keyId       the ID of the AWS KMS key for which to retrieve the policy
     * @param policyName the name of the key policy to retrieve
     * @return a {@link CompletableFuture} that, when completed, contains the key
policy as a {@link String}
     */
    public CompletableFuture<String> getKeyPolicyAsync(String keyId, String
policyName) {
        GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
```

```
            .keyId(keyId)
            .policyName(policyName)
            .build();

        return getAsyncClient().getKeyPolicy(policyRequest)
            .thenApply(response -> {
                String policy = response.policy();
                logger.info("The response is: " + policy);
                return policy;
            })
            .exceptionally(ex -> {
                throw new RuntimeException("Failed to get key policy", ex);
            });
    }

    /**
     * Asynchronously signs and verifies data using AWS KMS.
     *
     * <p>The method performs the following steps:
     * <ol>
     *     <li>Creates an AWS KMS key with the specified key spec, key usage, and
 origin.</li>
     *     <li>Signs the provided message using the created KMS key and the RSASSA-
PSS-SHA-256 algorithm.</li>
     *     <li>Verifies the signature of the message using the created KMS key and
 the RSASSA-PSS-SHA-256 algorithm.</li>
     * </ol>
     *
     * @return a {@link CompletableFuture} that completes with the result of the
 signature verification,
     *         {@code true} if the signature is valid, {@code false} otherwise.
     * @throws KmsException if any error occurs during the KMS operations.
     * @throws RuntimeException if an unexpected error occurs.
     */
    public CompletableFuture<Boolean> signVerifyDataAsync() {
        String signMessage = "Here is the message that will be digitally signed";

        // Create an AWS KMS key used to digitally sign data.
        CreateKeyRequest createKeyRequest = CreateKeyRequest.builder()
            .keySpec(KeySpec.RSA_2048)
            .keyUsage(KeyUsageType.SIGN_VERIFY)
            .origin(OriginType.AWS_KMS)
            .build();
```

```
        return getAsyncClient().createKey(createKeyRequest)
            .thenCompose(createKeyResponse -> {
                String keyId = createKeyResponse.keyMetadata().keyId();

                SdkBytes messageBytes = SdkBytes.fromString(signMessage,
Charset.defaultCharset());
                SignRequest signRequest = SignRequest.builder()
                    .keyId(keyId)
                    .message(messageBytes)
                    .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                    .build();

                return getAsyncClient().sign(signRequest)
                    .thenCompose(signResponse -> {
                        byte[] signedBytes = signResponse.signature().asByteArray();

                        VerifyRequest verifyRequest = VerifyRequest.builder()
                            .keyId(keyId)

.message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset())))

.signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))

.signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                            .build();

                        return getAsyncClient().verify(verifyRequest)
                            .thenApply(verifyResponse -> {
                                return (boolean) verifyResponse.signatureValid();
                            });
                    });
            })
            .exceptionally(throwable -> {
               throw new RuntimeException("Failed to sign or verify data",
throwable);
            });
    }

    /**
     * Asynchronously tags a KMS key with a specific tag.
     *
     * @param keyId the ID of the KMS key to be tagged
     * @return a {@link CompletableFuture} that completes when the tagging operation
is finished
```

```
     */
    public CompletableFuture<Void> tagKMSKeyAsync(String keyId) {
        Tag tag = Tag.builder()
            .tagKey("Environment")
            .tagValue("Production")
            .build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .keyId(keyId)
            .tags(tag)
            .build();

        return getAsyncClient().tagResource(tagResourceRequest)
            .thenRun(() -> {
                logger.info("{} key was tagged", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to tag the KMS key", throwable);
            });
    }

    /**
     * Deletes a specific KMS alias asynchronously.
     *
     * @param aliasName the name of the alias to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous operation
of deleting the specified alias
     */
    public CompletableFuture<Void> deleteSpecificAliasAsync(String aliasName) {
        DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
            .aliasName(aliasName)
            .build();

        return getAsyncClient().deleteAlias(deleteAliasRequest)
            .thenRun(() -> {
                logger.info("Alias {} has been deleted successfully", aliasName);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to delete alias: " + aliasName,
throwable);
            });
    }

    /**
```

```
    * Asynchronously disables the specified AWS Key Management Service (KMS) key.
    *
    * @param keyId the ID or Amazon Resource Name (ARN) of the KMS key to be
disabled
    * @return a CompletableFuture that, when completed, indicates that the key has
been disabled successfully
    */
   public CompletableFuture<Void> disableKeyAsync(String keyId) {
       DisableKeyRequest keyRequest = DisableKeyRequest.builder()
           .keyId(keyId)
           .build();

       return getAsyncClient().disableKey(keyRequest)
           .thenRun(() -> {
               logger.info("Key {} has been disabled successfully",keyId);
           })
           .exceptionally(throwable -> {
               throw new RuntimeException("Failed to disable key: " + keyId,
throwable);
           });
   }

   /**
    * Deletes a KMS key asynchronously.
    *
    * <p><strong>Warning:</strong> Deleting a KMS key is a destructive and
potentially dangerous operation.
    * When a KMS key is deleted, all data that was encrypted under the KMS key
becomes unrecoverable.
    * This means that any files, databases, or other data that were encrypted using
the deleted KMS key
    * will become permanently inaccessible. Exercise extreme caution when deleting
KMS keys.</p>
    *
    * @param keyId the ID of the KMS key to delete
    * @return a {@link CompletableFuture} that completes when the key deletion is
scheduled
    */
   public CompletableFuture<Void> deleteKeyAsync(String keyId) {
       ScheduleKeyDeletionRequest deletionRequest =
ScheduleKeyDeletionRequest.builder()
           .keyId(keyId)
           .pendingWindowInDays(7)
           .build();
```

```
        return getAsyncClient().scheduleKeyDeletion(deletionRequest)
            .thenRun(() -> {
                logger.info("Key {} will be deleted in 7 days", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to schedule key deletion for key
 ID: " + keyId, throwable);
            });
    }


    public String getAccountId(){
        try (StsClient stsClient = StsClient.create()){
            GetCallerIdentityResponse callerIdentity =
 stsClient.getCallerIdentity();
            return callerIdentity.account();
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - CreateAlias
  - CreateGrant
  - CreateKey
  - Decrypt
  - DescribeKey
  - DisableKey
  - EnableKey
  - Encrypt
  - GetKeyPolicy
  - ListAliases
  - ListGrants
  - ListKeys
  - RevokeGrant
  - ScheduleKeyDeletion

- Sign
- TagResource

操作

`CreateAlias`

以下代码示例演示了如何使用 `CreateAlias`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Creates a custom alias for the specified target key asynchronously.
    *
    * @param targetKeyId the ID of the target key for the alias
    * @param aliasName   the name of the alias to create
    * @return a {@link CompletableFuture} that completes when the alias creation
operation is finished
    */
   public CompletableFuture<Void> createCustomAliasAsync(String targetKeyId, String
aliasName) {
       CreateAliasRequest aliasRequest = CreateAliasRequest.builder()
           .aliasName(aliasName)
           .targetKeyId(targetKeyId)
           .build();

       CompletableFuture<CreateAliasResponse> responseFuture =
getAsyncClient().createAlias(aliasRequest);
       responseFuture.whenComplete((response, exception) -> {
           if (exception == null) {
               logger.info("{} was successfully created.", aliasName);
           } else {
               if (exception instanceof ResourceExistsException) {
                   logger.info("Alias [{}] already exists. Moving on...",
aliasName);
```

```
                } else if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while creating
alias: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred while
creating alias: " + exception.getMessage(), exception);
                }
            }
        });

        return responseFuture.thenApply(response -> null);
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateAlias中的。

`CreateGrant`

以下代码示例演示了如何使用 `CreateGrant`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Grants permissions to a specified principal on a customer master key (CMK)
asynchronously.
     *
     * @param keyId             The unique identifier for the customer master key
(CMK) that the grant applies to.
     * @param granteePrincipal  The principal that is given permission to perform
the operations that the grant permits on the CMK.
     * @return A {@link CompletableFuture} that, when completed, contains the ID of
the created grant.
     * @throws RuntimeException If an error occurs during the grant creation
process.
     */
```

```
   public CompletableFuture<String> grantKeyAsync(String keyId, String
granteePrincipal) {
       List<GrantOperation> grantPermissions = List.of(
           GrantOperation.ENCRYPT,
           GrantOperation.DECRYPT,
           GrantOperation.DESCRIBE_KEY
       );

       CreateGrantRequest grantRequest = CreateGrantRequest.builder()
           .keyId(keyId)
           .name("grant1")
           .granteePrincipal(granteePrincipal)
           .operations(grantPermissions)
           .build();

       CompletableFuture<CreateGrantResponse> responseFuture =
getAsyncClient().createGrant(grantRequest);
       responseFuture.whenComplete((response, ex) -> {
           if (ex == null) {
               logger.info("Grant created successfully with ID: " +
response.grantId());
           } else {
               if (ex instanceof KmsException kmsEx) {
                   throw new RuntimeException("Failed to create grant: " +
kmsEx.getMessage(), kmsEx);
               } else {
                   throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage(), ex);
               }
           }
       });

       return responseFuture.thenApply(CreateGrantResponse::grantId);
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateGrant中的。

CreateKey

以下代码示例演示了如何使用 CreateKey。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Creates a new symmetric encryption key asynchronously.
     *
     * @param keyDesc the description of the key to be created
     * @return a {@link CompletableFuture} that completes with the ID of the newly
created key
     * @throws RuntimeException if an error occurs while creating the key
     */
    public CompletableFuture<String> createKeyAsync(String keyDesc) {
        CreateKeyRequest keyRequest = CreateKeyRequest.builder()
            .description(keyDesc)
            .keySpec(KeySpec.SYMMETRIC_DEFAULT)
            .keyUsage(KeyUsageType.ENCRYPT_DECRYPT)
            .build();

        return getAsyncClient().createKey(keyRequest)
            .thenApply(resp -> resp.keyMetadata().keyId())
            .exceptionally(ex -> {
                throw new RuntimeException("An error occurred while creating the
key: " + ex.getMessage(), ex);
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateKey中的。

`Decrypt`

以下代码示例演示了如何使用 `Decrypt`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Asynchronously decrypts the given encrypted data using the specified key ID.
     *
     * @param encryptedData The encrypted data to be decrypted.
     * @param keyId The ID of the key to be used for decryption.
     * @return A CompletableFuture that, when completed, will contain the decrypted
data as a String.
     *         If an error occurs during the decryption process, the
CompletableFuture will complete
     *         exceptionally with the error, and the method will return an empty
String.
     */
    public CompletableFuture<String> decryptDataAsync(SdkBytes encryptedData, String
keyId) {
        DecryptRequest decryptRequest = DecryptRequest.builder()
            .ciphertextBlob(encryptedData)
            .keyId(keyId)
            .build();

        CompletableFuture<DecryptResponse> responseFuture =
getAsyncClient().decrypt(decryptRequest);
        responseFuture.whenComplete((decryptResponse, exception) -> {
            if (exception == null) {
                logger.info("Data decrypted successfully for key ID: " + keyId);
            } else {
                if (exception instanceof KmsException kmsEx) {
                    throw new RuntimeException("KMS error occurred while decrypting
data: " + kmsEx.getMessage(), kmsEx);
                } else {
                    throw new RuntimeException("An unexpected error occurred while
decrypting data: " + exception.getMessage(), exception);
                }
            }
        });
```

```
        return responseFuture.thenApply(decryptResponse ->
decryptResponse.plaintext().asString(StandardCharsets.UTF_8));
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 Decrypt。

`DeleteAlias`

以下代码示例演示了如何使用 `DeleteAlias`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Deletes a specific KMS alias asynchronously.
     *
     * @param aliasName the name of the alias to be deleted
     * @return a {@link CompletableFuture} representing the asynchronous operation
of deleting the specified alias
     */
    public CompletableFuture<Void> deleteSpecificAliasAsync(String aliasName) {
        DeleteAliasRequest deleteAliasRequest = DeleteAliasRequest.builder()
            .aliasName(aliasName)
            .build();

        return getAsyncClient().deleteAlias(deleteAliasRequest)
            .thenRun(() -> {
                logger.info("Alias {} has been deleted successfully", aliasName);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to delete alias: " + aliasName,
throwable);
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteAlias中的。

`DescribeKey`

以下代码示例演示了如何使用 `DescribeKey`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Asynchronously checks if a specified key is enabled.
    *
    * @param keyId the ID of the key to check
    * @return a {@link CompletableFuture} that, when completed, indicates whether
the key is enabled or not
    *
    * @throws RuntimeException if an exception occurs while checking the key state
    */
   public CompletableFuture<Boolean> isKeyEnabledAsync(String keyId) {
       DescribeKeyRequest keyRequest = DescribeKeyRequest.builder()
           .keyId(keyId)
           .build();

       CompletableFuture<DescribeKeyResponse> responseFuture =
getAsyncClient().describeKey(keyRequest);
       return responseFuture.whenComplete((resp, ex) -> {
           if (resp != null) {
               KeyState keyState = resp.keyMetadata().keyState();
               if (keyState == KeyState.ENABLED) {
                   logger.info("The key is enabled.");
               } else {
                   logger.info("The key is not enabled. Key state: {}", keyState);
               }
           } else {
```

```
                throw new RuntimeException(ex);
            }
        }).thenApply(resp -> resp.keyMetadata().keyState() == KeyState.ENABLED);
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeKey中的。

`DisableKey`

以下代码示例演示了如何使用 `DisableKey`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Asynchronously disables the specified AWS Key Management Service (KMS) key.
     *
     * @param keyId the ID or Amazon Resource Name (ARN) of the KMS key to be
disabled
     * @return a CompletableFuture that, when completed, indicates that the key has
been disabled successfully
     */
    public CompletableFuture<Void> disableKeyAsync(String keyId) {
        DisableKeyRequest keyRequest = DisableKeyRequest.builder()
            .keyId(keyId)
            .build();

        return getAsyncClient().disableKey(keyRequest)
            .thenRun(() -> {
                logger.info("Key {} has been disabled successfully",keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to disable key: " + keyId,
throwable);
            });
```

```
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DisableKey中的。

`EnableKey`

以下代码示例演示了如何使用 `EnableKey`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Asynchronously enables the specified key.
    *
    * @param keyId the ID of the key to enable
    * @return a {@link CompletableFuture} that completes when the key has been
enabled
    */
   public CompletableFuture<Void> enableKeyAsync(String keyId) {
       EnableKeyRequest enableKeyRequest = EnableKeyRequest.builder()
           .keyId(keyId)
           .build();

       CompletableFuture<EnableKeyResponse> responseFuture =
getAsyncClient().enableKey(enableKeyRequest);
       responseFuture.whenComplete((response, exception) -> {
           if (exception == null) {
               logger.info("Key with ID [{}] has been enabled.", keyId);
           } else {
               if (exception instanceof KmsException kmsEx) {
                   throw new RuntimeException("KMS error occurred while enabling
key: " + kmsEx.getMessage(), kmsEx);
               } else {
                   throw new RuntimeException("An unexpected error occurred while
enabling key: " + exception.getMessage(), exception);
               }
```

```
            }
        });

        return responseFuture.thenApply(response -> null);
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考EnableKey中的。

`Encrypt`

以下代码示例演示了如何使用 `Encrypt`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Encrypts the given text asynchronously using the specified KMS client and key
ID.
     *
     * @param keyId the ID of the KMS key to use for encryption
     * @param text the text to encrypt
     * @return a CompletableFuture that completes with the encrypted data as an
SdkBytes object
     */
    public CompletableFuture<SdkBytes> encryptDataAsync(String keyId, String text) {
        SdkBytes myBytes = SdkBytes.fromUtf8String(text);
        EncryptRequest encryptRequest = EncryptRequest.builder()
            .keyId(keyId)
            .plaintext(myBytes)
            .build();

        CompletableFuture<EncryptResponse> responseFuture =
getAsyncClient().encrypt(encryptRequest).toCompletableFuture();
        return responseFuture.whenComplete((response, ex) -> {
            if (response != null) {
                String algorithm = response.encryptionAlgorithm().toString();
```

```
                logger.info("The string was encrypted with algorithm {}.",
algorithm);
            } else {
                throw new RuntimeException(ex);
            }
        }).thenApply(EncryptResponse::ciphertextBlob);
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 Encrypt。

`ListAliases`

以下代码示例演示了如何使用 `ListAliases`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Asynchronously lists all the aliases in the current AWS account.
     *
     * @return a {@link CompletableFuture} that completes when the list of aliases
has been processed
     */
    public CompletableFuture<Object> listAllAliasesAsync() {
        ListAliasesRequest aliasesRequest = ListAliasesRequest.builder()
            .limit(15)
            .build();

        ListAliasesPublisher paginator =
getAsyncClient().listAliasesPaginator(aliasesRequest);
        return paginator.subscribe(response -> {
                response.aliases().forEach(alias ->
                    logger.info("The alias name is: " + alias.aliasName())
                );
            })
            .thenApply(v -> null)
```

```
            .exceptionally(ex -> {
                if (ex.getCause() instanceof KmsException) {
                    KmsException e = (KmsException) ex.getCause();
                    throw new RuntimeException("A KMS exception occurred: " +
e.getMessage());
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
ex.getMessage());
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListAliases中的。

`ListGrants`

以下代码示例演示了如何使用 `ListGrants`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Asynchronously displays the grant IDs for the specified key ID.
     *
     * @param keyId the ID of the AWS KMS key for which to list the grants
     * @return a {@link CompletableFuture} that, when completed, will be null if
the operation succeeded, or will throw a {@link RuntimeException} if the operation
failed
     * @throws RuntimeException if there was an error listing the grants, either due
to an {@link KmsException} or an unexpected error
     */
    public CompletableFuture<Object> displayGrantIdsAsync(String keyId) {
        ListGrantsRequest grantsRequest = ListGrantsRequest.builder()
            .keyId(keyId)
            .limit(15)
            .build();
```

```
        ListGrantsPublisher paginator =
getAsyncClient().listGrantsPaginator(grantsRequest);
        return paginator.subscribe(response -> {
                response.grants().forEach(grant -> {
                    logger.info("The grant Id is: " + grant.grantId());
                });
            })
            .thenApply(v -> null)
            .exceptionally(ex -> {
                Throwable cause = ex.getCause();
                if (cause instanceof KmsException) {
                    throw new RuntimeException("Failed to list grants: " +
cause.getMessage(), cause);
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
cause.getMessage(), cause);
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListGrants中的。

`ListKeyPolicies`

以下代码示例演示了如何使用 `ListKeyPolicies`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Asynchronously retrieves the key policy for the specified key ID and policy
name.
     *
     * @param keyId       the ID of the AWS KMS key for which to retrieve the policy
     * @param policyName the name of the key policy to retrieve
```

```
     * @return a {@link CompletableFuture} that, when completed, contains the key
 policy as a {@link String}
     */
    public CompletableFuture<String> getKeyPolicyAsync(String keyId, String
 policyName) {
        GetKeyPolicyRequest policyRequest = GetKeyPolicyRequest.builder()
            .keyId(keyId)
            .policyName(policyName)
            .build();

        return getAsyncClient().getKeyPolicy(policyRequest)
            .thenApply(response -> {
                String policy = response.policy();
                logger.info("The response is: " + policy);
                return policy;
            })
            .exceptionally(ex -> {
                throw new RuntimeException("Failed to get key policy", ex);
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ListKeyPolicies](#)中的。

`ListKeys`

以下代码示例演示了如何使用 `ListKeys`。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.services.kms.KmsAsyncClient;
import software.amazon.awssdk.services.kms.model.ListKeysRequest;
import software.amazon.awssdk.services.kms.paginators.ListKeysPublisher;
import java.util.concurrent.CompletableFuture;

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloKMS {
    public static void main(String[] args) {
        listAllKeys();
    }

    public static void listAllKeys() {
        KmsAsyncClient kmsAsyncClient = KmsAsyncClient.builder()
            .build();
        ListKeysRequest listKeysRequest = ListKeysRequest.builder()
            .limit(15)
            .build();

        /*
         * The `subscribe` method is required when using paginator methods in the
 AWS SDK
         * because paginator methods return an instance of a `ListKeysPublisher`,
 which is
         * based on a reactive stream. This allows asynchronous retrieval of
 paginated
         * results as they become available. By subscribing to the stream, we can
 process
         * each page of results as they are emitted.
         */
        ListKeysPublisher keysPublisher =
 kmsAsyncClient.listKeysPaginator(listKeysRequest);
        CompletableFuture<Void> future = keysPublisher
            .subscribe(r -> r.keys().forEach(key ->
                System.out.println("The key ARN is: " + key.keyArn() + ". The key Id
 is: " + key.keyId())))
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    System.err.println("Error occurred: " + exception.getMessage());
                } else {
                    System.out.println("Successfully listed all keys.");
                }
            });
```

```
        try {
            future.join();
        } catch (Exception e) {
            System.err.println("Failed to list keys: " + e.getMessage());
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListKeys中的。

`RevokeGrant`

以下代码示例演示了如何使用 `RevokeGrant`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Revokes a grant for the specified AWS KMS key asynchronously.
    *
    * @param keyId   The ID or key ARN of the AWS KMS key.
    * @param grantId The identifier of the grant to be revoked.
    * @return A {@link CompletableFuture} representing the asynchronous operation
of revoking the grant.
    *         The {@link CompletableFuture} will complete with a {@link
RevokeGrantResponse} object
    *         if the operation is successful, or with a {@code null} value if an
error occurs.
    */
   public CompletableFuture<RevokeGrantResponse> revokeKeyGrantAsync(String keyId,
String grantId) {
       RevokeGrantRequest grantRequest = RevokeGrantRequest.builder()
           .keyId(keyId)
           .grantId(grantId)
           .build();
```

```
        CompletableFuture<RevokeGrantResponse> responseFuture =
getAsyncClient().revokeGrant(grantRequest);
        responseFuture.whenComplete((response, exception) -> {
            if (exception == null) {
                logger.info("Grant ID: [" + grantId + "] was successfully
revoked!");
            } else {
                if (exception instanceof KmsException kmsEx) {
                    if (kmsEx.getMessage().contains("Grant does not exist")) {
                        logger.info("The grant ID '" + grantId + "' does not exist.
Moving on...");
                    } else {
                        throw new RuntimeException("KMS error occurred: " +
kmsEx.getMessage(), kmsEx);
                    }
                } else {
                    throw new RuntimeException("An unexpected error occurred: " +
exception.getMessage(), exception);
                }
            }
        });

        return responseFuture;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考RevokeGrant中的。

`ScheduleKeyDeletion`

以下代码示例演示了如何使用 `ScheduleKeyDeletion`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
```

```
    * Deletes a KMS key asynchronously.
    *
    * <p><strong>Warning:</strong> Deleting a KMS key is a destructive and
potentially dangerous operation.
    * When a KMS key is deleted, all data that was encrypted under the KMS key
becomes unrecoverable.
    * This means that any files, databases, or other data that were encrypted using
the deleted KMS key
    * will become permanently inaccessible. Exercise extreme caution when deleting
KMS keys.</p>
    *
    * @param keyId the ID of the KMS key to delete
    * @return a {@link CompletableFuture} that completes when the key deletion is
scheduled
    */
   public CompletableFuture<Void> deleteKeyAsync(String keyId) {
       ScheduleKeyDeletionRequest deletionRequest =
ScheduleKeyDeletionRequest.builder()
           .keyId(keyId)
           .pendingWindowInDays(7)
           .build();

       return getAsyncClient().scheduleKeyDeletion(deletionRequest)
           .thenRun(() -> {
               logger.info("Key {} will be deleted in 7 days", keyId);
           })
           .exceptionally(throwable -> {
               throw new RuntimeException("Failed to schedule key deletion for key
ID: " + keyId, throwable);
           });
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ScheduleKeyDeletion中的。

Sign

以下代码示例演示了如何使用 Sign。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Asynchronously signs and verifies data using AWS KMS.
     *
     * <p>The method performs the following steps:
     * <ol>
     *     <li>Creates an AWS KMS key with the specified key spec, key usage, and
 origin.</li>
     *     <li>Signs the provided message using the created KMS key and the RSASSA-
PSS-SHA-256 algorithm.</li>
     *     <li>Verifies the signature of the message using the created KMS key and
 the RSASSA-PSS-SHA-256 algorithm.</li>
     * </ol>
     *
     * @return a {@link CompletableFuture} that completes with the result of the
 signature verification,
     *         {@code true} if the signature is valid, {@code false} otherwise.
     * @throws KmsException if any error occurs during the KMS operations.
     * @throws RuntimeException if an unexpected error occurs.
     */
    public CompletableFuture<Boolean> signVerifyDataAsync() {
        String signMessage = "Here is the message that will be digitally signed";

        // Create an AWS KMS key used to digitally sign data.
        CreateKeyRequest createKeyRequest = CreateKeyRequest.builder()
            .keySpec(KeySpec.RSA_2048)
            .keyUsage(KeyUsageType.SIGN_VERIFY)
            .origin(OriginType.AWS_KMS)
            .build();

        return getAsyncClient().createKey(createKeyRequest)
            .thenCompose(createKeyResponse -> {
                String keyId = createKeyResponse.keyMetadata().keyId();
```

```
                SdkBytes messageBytes = SdkBytes.fromString(signMessage,
Charset.defaultCharset());
                SignRequest signRequest = SignRequest.builder()
                    .keyId(keyId)
                    .message(messageBytes)
                    .signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                    .build();

                return getAsyncClient().sign(signRequest)
                    .thenCompose(signResponse -> {
                        byte[] signedBytes = signResponse.signature().asByteArray();

                        VerifyRequest verifyRequest = VerifyRequest.builder()
                            .keyId(keyId)

.message(SdkBytes.fromByteArray(signMessage.getBytes(Charset.defaultCharset())))

.signature(SdkBytes.fromByteBuffer(ByteBuffer.wrap(signedBytes)))

.signingAlgorithm(SigningAlgorithmSpec.RSASSA_PSS_SHA_256)
                            .build();

                        return getAsyncClient().verify(verifyRequest)
                            .thenApply(verifyResponse -> {
                                return (boolean) verifyResponse.signatureValid();
                            });
                    });
            })
            .exceptionally(throwable -> {
               throw new RuntimeException("Failed to sign or verify data",
throwable);
            });
   }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 Sign。

TagResource

以下代码示例演示了如何使用 TagResource。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Asynchronously tags a KMS key with a specific tag.
     *
     * @param keyId the ID of the KMS key to be tagged
     * @return a {@link CompletableFuture} that completes when the tagging operation
is finished
     */
    public CompletableFuture<Void> tagKMSKeyAsync(String keyId) {
        Tag tag = Tag.builder()
            .tagKey("Environment")
            .tagValue("Production")
            .build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
            .keyId(keyId)
            .tags(tag)
            .build();

        return getAsyncClient().tagResource(tagResourceRequest)
            .thenRun(() -> {
                logger.info("{} key was tagged", keyId);
            })
            .exceptionally(throwable -> {
                throw new RuntimeException("Failed to tag the KMS key", throwable);
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考TagResource中的。

使用 SDK for Java 2.x 的 SDK 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

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

场景是向您展示如何通过在一个服务中调用多个函数或与其他 AWS 服务服务结合来完成特定任务的代码示例。

AWS 社区贡献就是由多个团队创建和维护的示例 AWS。要提供反馈，请使用链接存储库中提供的机制。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 基本功能
- 操作
- 场景
- 无服务器示例
- AWS 社区捐款

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。

适用于 Java 的 SDK 2.x

> **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。

```
/**
```

```
     * Lists the AWS Lambda functions associated with the current AWS account.
     *
     * @param awsLambda an instance of the {@link LambdaClient} class, which is used
to interact with the AWS Lambda service
     *
     * @throws LambdaException if an error occurs while interacting with the AWS
Lambda service
     */
    public static void listFunctions(LambdaClient awsLambda) {
        try {
            ListFunctionsResponse functionResult = awsLambda.listFunctions();
            List<FunctionConfiguration> list = functionResult.functions();
            for (FunctionConfiguration config : list) {
                System.out.println("The function name is " + config.functionName());
            }

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListFunctions中的。

基本功能

了解基本功能

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。

适用于 Java 的 SDK 2.x

> **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。

```
/*
 *  Lambda function names appear as:
 *
 *  arn:aws:lambda:us-west-2:335556666777:function:HelloFunction
 *
 *  To find this value, look at the function in the AWS Management Console.
 *
 *  Before running this Java code example, set up your development environment,
 including your credentials.
 *
 *  For more information, see this documentation topic:
 *
 *  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 *  This example performs the following tasks:
 *
 * 1. Creates an AWS Lambda function.
 * 2. Gets a specific AWS Lambda function.
 * 3. Lists all Lambda functions.
 * 4. Invokes a Lambda function.
 * 5. Updates the Lambda function code and invokes it again.
 * 6. Updates a Lambda function's configuration value.
 * 7. Deletes a Lambda function.
 */

public class LambdaScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) throws InterruptedException {
        final String usage = """

            Usage:
                <functionName> <role> <handler> <bucketName> <key>\s
```

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

        if (args.length != 5) {
              System.out.println(usage);
              return;
        }

        String functionName = args[0];
        String role = args[1];
        String handler = args[2];
        String bucketName = args[3];
        String key = args[4];
        LambdaClient awsLambda = LambdaClient.builder()
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS Lambda Basics scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create an AWS Lambda function.");
        String funArn = createLambdaFunction(awsLambda, functionName, key,
bucketName, role, handler);
        System.out.println("The AWS Lambda ARN is " + funArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Get the " + functionName + " AWS Lambda function.");
        getFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. List all AWS Lambda functions.");
        listFunctions(awsLambda);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Invoke the Lambda function.");
        System.out.println("*** Sleep for 1 min to get Lambda function ready.");
        Thread.sleep(60000);
        invokeFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Update the Lambda function code and invoke it
again.");
        updateFunctionCode(awsLambda, functionName, bucketName, key);
        System.out.println("*** Sleep for 1 min to get Lambda function ready.");
        Thread.sleep(60000);
        invokeFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Update a Lambda function's configuration value.");
        updateFunctionConfiguration(awsLambda, functionName, handler);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the AWS Lambda function.");
        LambdaScenario.deleteLambdaFunction(awsLambda, functionName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The AWS Lambda scenario completed successfully");
        System.out.println(DASHES);
        awsLambda.close();
    }

    /**
     * Creates a new Lambda function in AWS using the AWS Lambda Java API.
     *
     * @param awsLambda    the AWS Lambda client used to interact with the AWS
Lambda service
     * @param functionName the name of the Lambda function to create
     * @param key          the S3 key of the function code
     * @param bucketName   the name of the S3 bucket containing the function code
     * @param role         the IAM role to assign to the Lambda function
     * @param handler      the fully qualified class name of the function handler
```

```
     * @return the Amazon Resource Name (ARN) of the created Lambda function
     */
    public static String createLambdaFunction(LambdaClient awsLambda,
                                               String functionName,
                                               String key,
                                               String bucketName,
                                               String role,
                                               String handler) {

        try {
            LambdaWaiter waiter = awsLambda.waiter();
            FunctionCode code = FunctionCode.builder()
                .s3Key(key)
                .s3Bucket(bucketName)
                .build();

            CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
                .functionName(functionName)
                .description("Created by the Lambda Java API")
                .code(code)
                .handler(handler)
                .runtime(Runtime.JAVA17)
                .role(role)
                .build();

            // Create a Lambda function using a waiter
            CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
            GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
                .functionName(functionName)
                .build();
            WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            return functionResponse.functionArn();

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    /**
```

```
    * Retrieves information about an AWS Lambda function.
    *
    * @param awsLambda    an instance of the {@link LambdaClient} class, which is
used to interact with the AWS Lambda service
    * @param functionName the name of the AWS Lambda function to retrieve
information about
    */
   public static void getFunction(LambdaClient awsLambda, String functionName) {
       try {
           GetFunctionRequest functionRequest = GetFunctionRequest.builder()
               .functionName(functionName)
               .build();

           GetFunctionResponse response = awsLambda.getFunction(functionRequest);
           System.out.println("The runtime of this Lambda function is " +
response.configuration().runtime());

       } catch (LambdaException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }

   /**
    * Lists the AWS Lambda functions associated with the current AWS account.
    *
    * @param awsLambda an instance of the {@link LambdaClient} class, which is used
to interact with the AWS Lambda service
    *
    * @throws LambdaException if an error occurs while interacting with the AWS
Lambda service
    */
   public static void listFunctions(LambdaClient awsLambda) {
       try {
           ListFunctionsResponse functionResult = awsLambda.listFunctions();
           List<FunctionConfiguration> list = functionResult.functions();
           for (FunctionConfiguration config : list) {
               System.out.println("The function name is " + config.functionName());
           }

       } catch (LambdaException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
```

```
    }

    /**
     * Invokes a specific AWS Lambda function.
     *
     * @param awsLambda    an instance of {@link LambdaClient} to interact with the
AWS Lambda service
     * @param functionName the name of the AWS Lambda function to be invoked
     */
    public static void invokeFunction(LambdaClient awsLambda, String functionName) {
        InvokeResponse res;
        try {
            // Need a SdkBytes instance for the payload.
            JSONObject jsonObj = new JSONObject();
            jsonObj.put("inputValue", "2000");
            String json = jsonObj.toString();
            SdkBytes payload = SdkBytes.fromUtf8String(json);

            InvokeRequest request = InvokeRequest.builder()
                .functionName(functionName)
                .payload(payload)
                .build();

            res = awsLambda.invoke(request);
            String value = res.payload().asUtf8String();
            System.out.println(value);

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Updates the code for an AWS Lambda function.
     *
     * @param awsLambda  the AWS Lambda client
     * @param functionName the name of the Lambda function to update
     * @param bucketName the name of the S3 bucket where the function code is
located
     * @param key the key (file name) of the function code in the S3 bucket
     * @throws LambdaException if there is an error updating the function code
     */
```

```
   public static void updateFunctionCode(LambdaClient awsLambda, String
functionName, String bucketName, String key) {
       try {
           LambdaWaiter waiter = awsLambda.waiter();
           UpdateFunctionCodeRequest functionCodeRequest =
UpdateFunctionCodeRequest.builder()
               .functionName(functionName)
               .publish(true)
               .s3Bucket(bucketName)
               .s3Key(key)
               .build();

           UpdateFunctionCodeResponse response =
awsLambda.updateFunctionCode(functionCodeRequest);
           GetFunctionConfigurationRequest getFunctionConfigRequest =
GetFunctionConfigurationRequest.builder()
               .functionName(functionName)
               .build();

           WaiterResponse<GetFunctionConfigurationResponse> waiterResponse = waiter
               .waitUntilFunctionUpdated(getFunctionConfigRequest);
           waiterResponse.matched().response().ifPresent(System.out::println);
           System.out.println("The last modified value is " +
response.lastModified());

       } catch (LambdaException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }

   /**
    * Updates the configuration of an AWS Lambda function.
    *
    * @param awsLambda     the {@link LambdaClient} instance to use for the AWS
Lambda operation
    * @param functionName  the name of the AWS Lambda function to update
    * @param handler       the new handler for the AWS Lambda function
    *
    * @throws LambdaException if there is an error while updating the function
configuration
    */
   public static void updateFunctionConfiguration(LambdaClient awsLambda, String
functionName, String handler) {
```

```
        try {
            UpdateFunctionConfigurationRequest configurationRequest =
 UpdateFunctionConfigurationRequest.builder()
                .functionName(functionName)
                .handler(handler)
                .runtime(Runtime.JAVA17)
                .build();

            awsLambda.updateFunctionConfiguration(configurationRequest);

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Deletes an AWS Lambda function.
     *
     * @param awsLambda     an instance of the {@link LambdaClient} class, which is
 used to interact with the AWS Lambda service
     * @param functionName  the name of the Lambda function to be deleted
     *
     * @throws LambdaException if an error occurs while deleting the Lambda function
     */
    public static void deleteLambdaFunction(LambdaClient awsLambda, String
 functionName) {
        try {
            DeleteFunctionRequest request = DeleteFunctionRequest.builder()
                .functionName(functionName)
                .build();

            awsLambda.deleteFunction(request);
            System.out.println("The " + functionName + " function was deleted");

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。

- CreateFunction
- DeleteFunction
- GetFunction
- Invoke
- ListFunctions
- UpdateFunctionCode
- UpdateFunctionConfiguration

操作

`CreateFunction`

以下代码示例演示了如何使用 `CreateFunction`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Creates a new Lambda function in AWS using the AWS Lambda Java API.
     *
     * @param awsLambda    the AWS Lambda client used to interact with the AWS
Lambda service
     * @param functionName the name of the Lambda function to create
     * @param key          the S3 key of the function code
     * @param bucketName   the name of the S3 bucket containing the function code
     * @param role         the IAM role to assign to the Lambda function
     * @param handler      the fully qualified class name of the function handler
     * @return the Amazon Resource Name (ARN) of the created Lambda function
     */
    public static String createLambdaFunction(LambdaClient awsLambda,
                                               String functionName,
                                               String key,
                                               String bucketName,
```

```
                                            String role,
                                            String handler) {

        try {
            LambdaWaiter waiter = awsLambda.waiter();
            FunctionCode code = FunctionCode.builder()
                .s3Key(key)
                .s3Bucket(bucketName)
                .build();

            CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
                .functionName(functionName)
                .description("Created by the Lambda Java API")
                .code(code)
                .handler(handler)
                .runtime(Runtime.JAVA17)
                .role(role)
                .build();

            // Create a Lambda function using a waiter
            CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
            GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
                .functionName(functionName)
                .build();
            WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            return functionResponse.functionArn();

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateFunction中的。

DeleteFunction

以下代码示例演示了如何使用 DeleteFunction。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Deletes an AWS Lambda function.
    *
    * @param awsLambda     an instance of the {@link LambdaClient} class, which is
used to interact with the AWS Lambda service
    * @param functionName  the name of the Lambda function to be deleted
    *
    * @throws LambdaException if an error occurs while deleting the Lambda function
    */
   public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName) {
       try {
           DeleteFunctionRequest request = DeleteFunctionRequest.builder()
               .functionName(functionName)
               .build();

           awsLambda.deleteFunction(request);
           System.out.println("The " + functionName + " function was deleted");

       } catch (LambdaException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteFunction中的。

`GetFunction`

以下代码示例演示了如何使用 `GetFunction`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Retrieves information about an AWS Lambda function.
     *
     * @param awsLambda    an instance of the {@link LambdaClient} class, which is
used to interact with the AWS Lambda service
     * @param functionName the name of the AWS Lambda function to retrieve
information about
     */
    public static void getFunction(LambdaClient awsLambda, String functionName) {
        try {
            GetFunctionRequest functionRequest = GetFunctionRequest.builder()
                .functionName(functionName)
                .build();

            GetFunctionResponse response = awsLambda.getFunction(functionRequest);
            System.out.println("The runtime of this Lambda function is " +
response.configuration().runtime());

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetFunction中的。

Invoke

以下代码示例演示了如何使用 Invoke。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Invokes a specific AWS Lambda function.
     *
     * @param awsLambda    an instance of {@link LambdaClient} to interact with the
AWS Lambda service
     * @param functionName the name of the AWS Lambda function to be invoked
     */
    public static void invokeFunction(LambdaClient awsLambda, String functionName) {
        InvokeResponse res;
        try {
            // Need a SdkBytes instance for the payload.
            JSONObject jsonObj = new JSONObject();
            jsonObj.put("inputValue", "2000");
            String json = jsonObj.toString();
            SdkBytes payload = SdkBytes.fromUtf8String(json);

            InvokeRequest request = InvokeRequest.builder()
                .functionName(functionName)
                .payload(payload)
                .build();

            res = awsLambda.invoke(request);
            String value = res.payload().asUtf8String();
            System.out.println(value);

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的 Invoke。

UpdateFunctionCode

以下代码示例演示了如何使用 UpdateFunctionCode。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Updates the code for an AWS Lambda function.
    *
    * @param awsLambda  the AWS Lambda client
    * @param functionName the name of the Lambda function to update
    * @param bucketName the name of the S3 bucket where the function code is
located
    * @param key the key (file name) of the function code in the S3 bucket
    * @throws LambdaException if there is an error updating the function code
    */
   public static void updateFunctionCode(LambdaClient awsLambda, String
functionName, String bucketName, String key) {
       try {
           LambdaWaiter waiter = awsLambda.waiter();
           UpdateFunctionCodeRequest functionCodeRequest =
UpdateFunctionCodeRequest.builder()
                   .functionName(functionName)
                   .publish(true)
                   .s3Bucket(bucketName)
                   .s3Key(key)
                   .build();

           UpdateFunctionCodeResponse response =
awsLambda.updateFunctionCode(functionCodeRequest);
           GetFunctionConfigurationRequest getFunctionConfigRequest =
GetFunctionConfigurationRequest.builder()
                   .functionName(functionName)
                   .build();

           WaiterResponse<GetFunctionConfigurationResponse> waiterResponse = waiter
                   .waitUntilFunctionUpdated(getFunctionConfigRequest);
```

```
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("The last modified value is " +
response.lastModified());

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考UpdateFunctionCode中的。

`UpdateFunctionConfiguration`

以下代码示例演示了如何使用 `UpdateFunctionConfiguration`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Updates the configuration of an AWS Lambda function.
     *
     * @param awsLambda     the {@link LambdaClient} instance to use for the AWS
Lambda operation
     * @param functionName  the name of the AWS Lambda function to update
     * @param handler       the new handler for the AWS Lambda function
     *
     * @throws LambdaException if there is an error while updating the function
configuration
     */
    public static void updateFunctionConfiguration(LambdaClient awsLambda, String
functionName, String handler) {
        try {
            UpdateFunctionConfigurationRequest configurationRequest =
UpdateFunctionConfigurationRequest.builder()
                .functionName(functionName)
```

```
                .handler(handler)
                .runtime(Runtime.JAVA17)
                .build();

            awsLambda.updateFunctionConfiguration(configurationRequest);

        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[UpdateFunctionConfiguration](#)中的。

场景

创建无服务器应用程序来管理照片

以下代码示例演示如何创建无服务器应用程序，让用户能够使用标签管理照片。

适用于 Java 的 SDK 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 Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

创建用于分析客户反馈的应用程序

以下代码示例说明如何创建应用程序来分析客户意见卡、翻译其母语、确定其情绪并根据译后的文本生成音频文件。

适用于 Java 的 SDK 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 API Gateway 调用 Lambda 函数

以下代码示例展示了如何创建由 Amazon API Gateway 调用的 AWS Lambda 函数。

适用于 Java 的 SDK 2.x

演示如何使用 Lambda Java 运行时 API 创建 AWS Lambda 函数。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通过 Amazon API Gateway 调用的 Lambda 函数，该函数扫描 Amazon DynamoDB 表获取工作周年纪念日，并使用 Amazon Simple Notification Service (Amazon SNS)向员工发送文本消息，祝贺他们的周年纪念日。

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 Gateway
- DynamoDB
- Lambda
- Amazon SNS

使用 Step Functions 调用 Lambda 函数

以下代码示例说明如何创建按顺序调用 AWS Lambda 函数的 AWS Step Functions 状态机。

适用于 Java 的 SDK 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
- Amazon SES
- Step Functions

使用计划的事件调用 Lambda 函数

以下代码示例显示如何创建由 Amazon EventBridge 计划事件调用的 AWS Lambda 函数。

适用于 Java 的 SDK 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。在本示例中，您使用 Lambda Java 运行时 API 创建 Lambda 函数。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

- CloudWatch 日志

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

使用 Neptune API 查询图形数据

以下代码示例展示了如何使用 Neptune API 查询图形数据。

适用于 Java 的 SDK 2.x

展示如何使用 Amazon Neptune Java API 创建查询 VPC 内的图形数据的 Lambda 函数。

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

- Lambda
- Neptune

无服务器示例

使用 Lambda 函数连接到 Amazon RDS 数据库

以下代码示例显示如何实现连接到 RDS 数据库的 Lambda 函数。该函数发出一个简单的数据库请求并返回结果。

适用于 Java 的 SDK 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。

在 Lambda 函数中使用 Java 连接到 Amazon 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。

适用于 Java 的 SDK 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一个 Lambda 函数，该函数接收通过接收来自 DynamoDB 流的记录而触发的事件。该函数检索 DynamoDB 有效负载，并记录下记录内容。

适用于 Java 的 SDK 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 DocumentDB 触发器调用 Lambda 函数

以下代码示例演示如何实现一个 Lambda 函数，该函数接收通过接收来自 DocumentDB 更改流的记录而触发的事件。该函数检索 DocumentDB 有效负载，并记录下记录内容。

适用于 Java 的 SDK 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 Amazon DocumentDB 事件与 Lambda 结合使用。

```
import java.util.List;
import java.util.Map;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;

public class Example implements RequestHandler<Map<String, Object>, String> {

    @SuppressWarnings("unchecked")
    @Override
    public String handleRequest(Map<String, Object> event, Context context) {
        List<Map<String, Object>> events = (List<Map<String, Object>>)
 event.get("events");
        for (Map<String, Object> record : events) {
            Map<String, Object> eventData = (Map<String, Object>)
 record.get("event");
            processEventData(eventData);
        }

        return "OK";
    }

    @SuppressWarnings("unchecked")
    private void processEventData(Map<String, Object> eventData) {
        String operationType = (String) eventData.get("operationType");
```

```
        System.out.println("operationType: %s".formatted(operationType));

        Map<String, Object> ns = (Map<String, Object>) eventData.get("ns");

        String db = (String) ns.get("db");
        System.out.println("db: %s".formatted(db));
        String coll = (String) ns.get("coll");
        System.out.println("coll: %s".formatted(coll));

        Map<String, Object> fullDocument = (Map<String, Object>)
 eventData.get("fullDocument");
        System.out.println("fullDocument: %s".formatted(fullDocument));
    }

}
```

通过 Amazon MSK 触发器调用 Lambda 函数

以下代码示例演示如何实现一个 Lambda 函数，该函数接收通过接收来自 Amazon MSK 集群的记录而触发的事件。该函数检索 MSK 有效负载，并记录下记录内容。

适用于 Java 的 SDK 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。

通过 Java 将 Amazon MSK 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.KafkaEvent;
import com.amazonaws.services.lambda.runtime.events.KafkaEvent.KafkaEventRecord;

import java.util.Base64;
import java.util.Map;

public class Example implements RequestHandler<KafkaEvent, Void> {
```

```
    @Override
    public Void handleRequest(KafkaEvent event, Context context) {
        for (Map.Entry<String, java.util.List<KafkaEventRecord>> entry :
 event.getRecords().entrySet()) {
            String key = entry.getKey();
            System.out.println("Key: " + key);

            for (KafkaEventRecord record : entry.getValue()) {
                System.out.println("Record: " + record);

                byte[] value = Base64.getDecoder().decode(record.getValue());
                String message = new String(value);
                System.out.println("Message: " + message);
            }
        }

        return null;
    }
}
```

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。

适用于 Java 的 SDK 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);
    }
```

```
}
```

通过 Amazon SNS 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 SNS 主题的消息而触发的事件。该函数从事件参数检索消息并记录每条消息的内容。

适用于 Java 的 SDK 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。

通过 Java 将 SNS 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.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);
        }
    }

}
```

通过 Amazon SQS 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 SNS 队列的消息而触发的事件。该函数从事件参数检索消息并记录每条消息的内容。

适用于 Java 的 SDK 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。

通过 Java 将 SQS 事件与 Lambda 结合使用。

```
// 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。

适用于 Java 的 SDK 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。

适用于 Java 的 SDK 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.util.ArrayList;
import java.util.List;

public class ProcessDynamodbRecords implements RequestHandler<DynamodbEvent,
 StreamsEventResponse> {

    @Override
    public StreamsEventResponse handleRequest(DynamodbEvent input, Context context)
 {

        List<StreamsEventResponse.BatchItemFailure> batchItemFailures = new
 ArrayList<>();
        String curRecordSequenceNumber = "";

        for (DynamodbEvent.DynamodbStreamRecord dynamodbStreamRecord :
 input.getRecords()) {
          try {
                //Process your record
                StreamRecord dynamodbRecord = dynamodbStreamRecord.getDynamodb();
```

```
                curRecordSequenceNumber = dynamodbRecord.getSequenceNumber();

            } catch (Exception e) {
                /* Since we are working with streams, we can return the failed item
 immediately.
                   Lambda will immediately begin to retry processing from this
 failed item onwards. */
                batchItemFailures.add(new
 StreamsEventResponse.BatchItemFailure(curRecordSequenceNumber));
                return new StreamsEventResponse(batchItemFailures);
            }
        }

       return new StreamsEventResponse();
    }
}
```

报告使用 Amazon SQS 触发器进行 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 SQS 队列的事件的 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。

适用于 Java 的 SDK 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 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>();

         for (SQSEvent.SQSMessage message : sqsEvent.getRecords()) {
             try {
                 //process your message
             } catch (Exception e) {
                 //Add failed message identifier to the batchItemFailures list
                 batchItemFailures.add(new
 SQSBatchResponse.BatchItemFailure(message.getMessageId()));
             }
         }
         return new SQSBatchResponse(batchItemFailures);
     }
}
```

AWS 社区捐款

构建和测试无服务器应用程序

以下代码示例演示如何使用 API Gateway 与 Lambda 和 DynamoDB 构建和测试无服务器应用程序

适用于 Java 的 SDK 2.x

演示如何使用 Java SDK 构建和测试包含 API Gateway 以及 Lambda 和 DynamoDB 的无服务器应用程序。

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

- API Gateway
- DynamoDB
- Lambda

使用适用于 Java 的 SDK 2.x 的 Amazon Lex 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 Lex 配合使用来执行操作和实现常见场景。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 场景

场景

构建 Amazon Lex 聊天机器人

以下代码示例演示如何创建用于吸引网站访客的聊天机器人。

适用于 Java 的 SDK 2.x

展示如何使用 Amazon Lex API 在 Web 应用程序中创建聊天机器人，以吸引网站访客。

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

使用适用于 Java 的 SDK 2.x 的 Amazon Location 示例

以下代码示例向您展示了如何使用 with Amazon Location 来执行操作和实现常见场景。 AWS SDK for Java 2.x

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 基本功能
- 操作

开始使用

Hello Amazon Location

以下代码示例显示了如何开始使用 Amazon Location Service。

适用于 Java 的 SDK 2.x

> **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。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, you need to create a collection using the AWS Management
 * console. For information, see the following documentation.
 *
 * https://docs.aws.amazon.com/location/latest/developerguide/geofence-gs.html

 */
public class HelloLocation {
```

```
   private static LocationAsyncClient locationAsyncClient;
   private static final Logger logger =
LoggerFactory.getLogger(HelloLocation.class);

   // This Singleton pattern ensures that only one `LocationClient`
   // instance.
   private static LocationAsyncClient getClient() {
       if (locationAsyncClient == null) {
           SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
               .maxConcurrency(100)
               .connectionTimeout(Duration.ofSeconds(60))
               .readTimeout(Duration.ofSeconds(60))
               .writeTimeout(Duration.ofSeconds(60))
               .build();

           ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
               .apiCallTimeout(Duration.ofMinutes(2))
               .apiCallAttemptTimeout(Duration.ofSeconds(90))
               .retryStrategy(RetryMode.STANDARD)
               .build();

           locationAsyncClient = LocationAsyncClient.builder()
               .httpClient(httpClient)
               .overrideConfiguration(overrideConfig)
               .build();
       }
       return locationAsyncClient;
   }

   public static void main(String[] args) {
       final String usage = """

           Usage:
               <collectionName>

           Where:
               collectionName - The Amazon location collection name.
           """;

       if (args.length != 1) {
           System.out.println(usage);
           System.exit(1);
```

```
        }

        String collectionName = args[0];
        listGeofences(collectionName);
    }

    /**
     * Lists geofences from a specified geofence collection asynchronously.
     *
     * @param collectionName The name of the geofence collection to list geofences
 from.
     * @return A {@link CompletableFuture} representing the result of the
 asynchronous operation.
     *         The future completes when all geofences have been processed and
 logged.
     */
    public static CompletableFuture<Void> listGeofences(String collectionName) {
        ListGeofencesRequest geofencesRequest = ListGeofencesRequest.builder()
                .collectionName(collectionName)
                .build();

        ListGeofencesPublisher paginator =
 getClient().listGeofencesPaginator(geofencesRequest);
        CompletableFuture<Void> future = paginator.subscribe(response -> {
            if (response.entries().isEmpty()) {
                logger.info("No Geofences were found in the collection.");
            } else {
                response.entries().forEach(geofence ->
                        logger.info("Geofence ID: " + geofence.geofenceId())
                );
            }
        });
        return future;
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - ListGeofenceCollections
  - ListGeofences

基本功能

了解基本功能

以下代码示例展示了如何：

- 创建 Amazon Location 地图。
- 创建 Amazon Location API 密钥。
- 显示地图 URL。
- 创建地理围栏集合。
- 存储地理围栏几何结构。
- 创建跟踪器资源。
- 更新设备的位置。
- 检索指定设备的最新位置更新。
- 创建路线计算器。
- 确定西雅图和温哥华之间的距离。
- 使用更高级别的 Amazon 位置 APIs。
- 删除 Amazon Location 资产。

适用于 Java 的 SDK 2.x

> **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。

运行一个交互式场景，演示 Amazon Location Service 的功能。

```
/*
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
```

```
 */
public class LocationScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    private static final Logger logger =
 LoggerFactory.getLogger(LocationScenario.class);
    static Scanner scanner = new Scanner(System.in);

    static LocationActions locationActions = new LocationActions();

    public static void main(String[] args) {
        final String usage = """

            Usage:    <mapName> <keyName> <collectionName> <geoId> <trackerName>
 <calculatorName> <deviceId>

            Where:
              mapName - The name of the map to be create (e.g., "AWSMap").
              keyName - The name of the API key to create (e.g., "AWSApiKey").
              collectionName - The name of the geofence collection (e.g.,
 "AWSLocationCollection").
              geoId - The geographic identifier used for the geofence or map (e.g.,
 "geoId").
              trackerName - The name of the tracker (e.g., "geoTracker").
              calculatorName - The name of the route calculator (e.g.,
 "AWSRouteCalc").
              deviceId - The ID of the device (e.g., "iPhone-112356").
            """;

        if (args.length != 7) {
            logger.info(usage);
            return;
        }

        String mapName = args[0];
        String keyName = args[1];
        String collectionName = args[2];
        String geoId = args[3];
        String trackerName = args[4];
        String calculatorName = args[5];
        String deviceId = args[6];

        logger.info("""
```

```
            AWS Location Service is a fully managed service offered by Amazon Web
Services (AWS) that
            provides location-based services for developers. This service simplifies
            the integration of location-based features into applications, making it
            easier to build and deploy location-aware applications.

            The AWS Location Service offers a range of location-based services,
            including:

            Maps: The service provides access to high-quality maps, satellite
imagery,\s
            and geospatial data from various providers, allowing developers to\s
            easily embed maps into their applications.

            Tracking: The Location Service enables real-time tracking of mobile
devices,\s
            assets, or other entities, allowing developers to build applications\s
            that can monitor the location of people, vehicles, or other objects.

            Geocoding: The service provides the ability to convert addresses or\s
            location names into geographic coordinates (latitude and longitude),\s
            and vice versa, enabling developers to integrate location-based search\s
            and routing functionality into their applications.
            """);
        waitForInputToContinue(scanner);
        try {
            runScenario(mapName, keyName, collectionName, geoId, trackerName,
calculatorName, deviceId);
        } catch (RuntimeException e) {
            // Clean up AWS Resources.
            cleanUp(mapName, keyName, collectionName, trackerName, calculatorName);
            logger.info(e.getMessage());
        }
    }

    public static void runScenario(String mapName, String keyName, String
collectionName, String geoId, String trackerName, String calculatorName, String
deviceId) {
        logger.info(DASHES);
        logger.info("1. Create a map");
        logger.info("""
             An AWS Location map can enhance the user experience of your
             application by providing accurate and personalized location-based
             features. For example, you could use the geocoding capabilities to
```

```
             allow users to search for and locate businesses, landmarks, or
             other points of interest within a specific region.
            """);

        waitForInputToContinue(scanner);
        String mapArn;
        try {
            mapArn = locationActions.createMap(mapName).join();
            logger.info("The Map ARN is: {}", mapArn);  // Log success in calling
code
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ServiceQuotaExceededException) {
                logger.error("The request exceeded the maximum quota: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred while creating the map.",
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Create an AWS Location API key");
        logger.info("""
            When you embed a map in a web app or website, the API key is
            included in the map tile URL to authenticate requests. You can
            restrict API keys to specific AWS Location operations (e.g., only
            maps, not geocoding). API keys can expire, ensuring temporary
            access control.
            """);

        try {
            String keyArn = locationActions.createKey(keyName, mapArn).join();
            logger.info("The API key was successfully created: {}", keyArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof AccessDeniedException) {
                logger.error("Request was denied: {}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred while creating the API
key.", cause);
```

```
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Display Map URL");
        logger.info("""
            In order to get the MAP URL, you need to get the API Key value.
            You can get the key value using the AWS Management Console under
            Location Services. This operation cannot be completed using the
            AWS SDK. For more information about getting the key value, see
            the AWS Location Documentation.
            """);
        String mapUrl = "https://maps.geo.aws.amazon.com/maps/v0/maps/"+mapName+"/
tiles/{z}/{x}/{y}?key={KeyValue}";
        logger.info("Embed this URL in your Web app: " + mapUrl);
        logger.info("");
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Create a geofence collection, which manages and stores
 geofences.");
        waitForInputToContinue(scanner);
        try {
            String collectionArn =
 locationActions.createGeofenceCollection(collectionName).join();
            logger.info("The geofence collection was successfully created: {}",
 collectionArn);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ConflictException) {
                logger.error("A conflict occurred: {}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred while creating the
 geofence collection.", cause);
            }
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);
```

```
        logger.info(DASHES);
        logger.info("5. Store a geofence geometry in a given geofence collection.");
        logger.info("""
            An AWS Location geofence is a virtual boundary that defines a geographic
area
            on a map. It is a useful feature for tracking the location of
            assets or monitoring the movement of objects within a specific region.

            To define a geofence, you need to specify the coordinates of a
            polygon that represents the area of interest. The polygon must be
            defined in a counter-clockwise direction, meaning that the points of
            the polygon must be listed in a counter-clockwise order.

            This is a requirement for the AWS Location service to correctly
            interpret the geofence and ensure that the location data is
            accurately processed within the defined area.
            """);

        waitForInputToContinue(scanner);
        try {
            locationActions.putGeofence(collectionName, geoId).join();
            logger.info("Successfully created geofence: {}", geoId);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ValidationException) {
                logger.error("A validation error occurred while creating geofence:
{}", cause.getMessage());
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("6. Create a tracker resource which lets you retrieve current
and historical location of devices..");
        waitForInputToContinue(scanner);
        try {
            String trackerArn = locationActions.createTracker(trackerName).join();
            logger.info("Successfully created tracker. ARN: {}", trackerArn);  //
Log success
```

```
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ConflictException) {
                logger.error("A conflict occurred while creating the tracker: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Update the position of a device in the location tracking
system.");
        logger.info("""
            The AWS location service does not enforce a strict format for deviceId,
but it must:
              - Be a string (case-sensitive).
              - Be 1-100 characters long.
              - Contain only:
                - Alphanumeric characters (A-Z, a-z, 0-9)
                - Underscores (_)
                - Hyphens (-)
                - Be the same ID used when sending and retrieving positions.
            """);

        waitForInputToContinue(scanner);
        try {
            CompletableFuture<BatchUpdateDevicePositionResponse> future =
locationActions.updateDevicePosition(trackerName, deviceId);
            future.join();
            logger.info(deviceId + " was successfully updated in the location
tracking system.");
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The resource was not found: {}", cause.getMessage(),
cause);
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage(),
cause);
```

```
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("8. Retrieve the most recent position update for a specified
device..");
        waitForInputToContinue(scanner);
        try {
            GetDevicePositionResponse response =
locationActions.getDevicePosition(trackerName, deviceId).join();
            logger.info("Successfully fetched device position: {}",
response.position());
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The resource was not found: {}", cause.getMessage(),
cause);
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("9. Create a route calculator.");
        waitForInputToContinue(scanner);
        try {
            CreateRouteCalculatorResponse response =
locationActions.createRouteCalculator(calculatorName).join();
            logger.info("Route calculator created successfully: {}",
response.calculatorArn());
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ConflictException) {
                logger.info("A conflict occurred: {}", cause.getMessage(), cause);
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
```

```
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("10. Determine the distance between Seattle and Vancouver using
the route calculator.");
        waitForInputToContinue(scanner);
        try {
            CalculateRouteResponse response =
locationActions.calcDistanceAsync(calculatorName).join();
            logger.info("Successfully calculated route. The distance in kilometers
is {}", response.summary().distance());
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The resource was not found: {}", cause.getMessage(),
cause);
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info("11. Use the GeoPlacesAsyncClient to perform additional
operations.");
        logger.info("""
            This scenario will show use of the GeoPlacesClient that enables
            location search and geocoding capabilities for your applications.\s

            We are going to use this client to perform these AWS Location tasks:
             - Reverse Geocoding (reverseGeocode): Converts geographic coordinates
into addresses.
             - Place Search (searchText): Finds places based on search queries.
             - Nearby Search (searchNearby): Finds places near a specific location.
            """);

        logger.info("First we will perform a Reverse Geocoding operation");
        waitForInputToContinue(scanner);
        try {
            locationActions.reverseGeocode().join();
```

```
            logger.info("Now we are going to perform a text search using coffee
shop.");
            waitForInputToContinue(scanner);
            locationActions.searchText("coffee shop").join();
            waitForInputToContinue(scanner);

            logger.info("Now we are going to perform a nearby Search.");
            //waitForInputToContinue(scanner);
            locationActions.searchNearBy().join();
            waitForInputToContinue(scanner);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof
software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                logger.error("A validation error occurred: {}", cause.getMessage(),
cause);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
cause);
            }
            return;
        }
        logger.info(DASHES);

        logger.info("12. Delete the AWS Location Services resources.");
        logger.info("Would you like to delete the AWS Location Services resources?
(y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            cleanUp(mapName, keyName, collectionName, trackerName, calculatorName);
        } else {
            logger.info("The AWS resources will not be deleted.");
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info(" This concludes the AWS Location Service scenario.");
        logger.info(DASHES);
    }

    /**
     * Cleans up resources by deleting the specified map, key, geofence collection,
tracker, and route calculator.
```

```
     *
     * @param mapName The name of the map to delete.
     * @param keyName The name of the key to delete.
     * @param collectionName The name of the geofence collection to delete.
     * @param trackerName The name of the tracker to delete.
     * @param calculatorName The name of the route calculator to delete.
     */
    private static void cleanUp(String mapName, String keyName, String
 collectionName, String trackerName, String calculatorName) {
        try {
            locationActions.deleteMap(mapName).join();
            locationActions.deleteKey(keyName).join();
            locationActions.deleteGeofenceCollectionAsync(collectionName).join();
            locationActions.deleteTracker(trackerName).join();
            locationActions.deleteRouteCalculator(calculatorName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.info("The resource was not found: {}", cause.getMessage(),
 cause);
            } else {
                logger.info("An unexpected error occurred: {}", cause.getMessage(),
 cause);
            }
            return;
        }
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Amazon Location Service SDK 方法的包装器类。

```
public class LocationActions {

    private static LocationAsyncClient locationAsyncClient;

    private static GeoPlacesAsyncClient geoPlacesAsyncClient;
    private static final Logger logger =
 LoggerFactory.getLogger(LocationActions.class);

    // This Singleton pattern ensures that only one `LocationClient`
    // instance is used throughout the application.
    private LocationAsyncClient getClient() {
        if (locationAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            locationAsyncClient = LocationAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return locationAsyncClient;
    }

    private static GeoPlacesAsyncClient getGeoPlacesClient() {
        if (geoPlacesAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
```

```
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            geoPlacesAsyncClient = GeoPlacesAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return geoPlacesAsyncClient;
    }


    /**
     * Performs a nearby places search based on the provided geographic coordinates
 (latitude and longitude).
     * The method sends an asynchronous request to search for places within a 1-
kilometer radius of the specified location.
     * The results are processed and printed once the search completes successfully.
     */
    public CompletableFuture<SearchNearbyResponse> searchNearBy() {
        double latitude = 37.7749;  // San Francisco
        double longitude = -122.4194;
        List<Double> queryPosition = List.of(longitude, latitude);

        // Set up the request for searching nearby places.
        SearchNearbyRequest request = SearchNearbyRequest.builder()
            .queryPosition(queryPosition)  // Set the position
            .queryRadius(1000L)  // Radius in meters (1000 meters = 1 km).
            .build();

        return getGeoPlacesClient().searchNearby(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof
 software.amazon.awssdk.services.geoplaces.model.ValidationException) {
```

```
                        throw new CompletionException("A validation error occurred:
" + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error performing place search",
exception);
                }

                // Process the response and print the results.
                response.resultItems().forEach(result -> {
                    logger.info("Place Name: " + result.placeType().name());
                    logger.info("Address: " + result.address().label());
                    logger.info("Distance: " + result.distance() + " meters");
                    logger.info("-------------------------");
                });
            });
    }


    /**
     * Searches for a place using the provided search query and prints the detailed
information of the first result.
     *
     * @param searchQuery the search query to be used for the place search (ex,
coffee shop)
     */
    public CompletableFuture<Void> searchText(String searchQuery) {
        double latitude = 37.7749;  // San Francisco
        double longitude = -122.4194;
        List<Double> queryPosition = List.of(longitude, latitude);

        SearchTextRequest request = SearchTextRequest.builder()
                .queryText(searchQuery)
                .biasPosition(queryPosition)
                .build();

        return getGeoPlacesClient().searchText(request)
                .thenCompose(response -> {
                    if (response.resultItems().isEmpty()) {
                        logger.info("No places found.");
                        return CompletableFuture.completedFuture(null);
                    }

                    // Get the first place ID
                    String placeId = response.resultItems().get(0).placeId();
```

```
                    logger.info("Found Place with id: " + placeId);

                    // Fetch detailed info using getPlace
                    GetPlaceRequest getPlaceRequest = GetPlaceRequest.builder()
                            .placeId(placeId)
                            .build();

                    return getGeoPlacesClient().getPlace(getPlaceRequest)
                            .thenAccept(placeResponse -> {
                                logger.info("Detailed Place Information:");
                                logger.info("Name: " +
 placeResponse.placeType().name());
                                logger.info("Address: " +
 placeResponse.address().label());

                                if (placeResponse.foodTypes() != null && !
placeResponse.foodTypes().isEmpty()) {
                                    logger.info("Food Types:");
                                    placeResponse.foodTypes().forEach(foodType -> {
                                        logger.info("  - " + foodType);
                                    });
                                } else {
                                    logger.info("No food types available.");
                                }
                                logger.info("-------------------------");
                            });
                })
                .exceptionally(exception -> {
                    Throwable cause = exception.getCause();
                    if (cause instanceof
 software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                        throw new CompletionException("A validation error occurred:
 " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error performing place search",
 exception);
                });
    }



    /**
     * Performs reverse geocoding using the AWS Geo Places API.
```

```
    * Reverse geocoding is the process of converting geographic coordinates
(latitude and longitude) to a human-readable address.
    * This method uses the latitude and longitude of San Francisco as the input,
and prints the resulting address.
    */
   public CompletableFuture<ReverseGeocodeResponse> reverseGeocode() {
       double latitude = 37.7749;  // San Francisco
       double longitude = -122.4194;
       logger.info("Use latitude 37.7749 and longitude -122.4194");

       // AWS expects [longitude, latitude].
       List<Double> queryPosition = List.of(longitude, latitude);
       ReverseGeocodeRequest request = ReverseGeocodeRequest.builder()
           .queryPosition(queryPosition)
           .build();
       CompletableFuture<ReverseGeocodeResponse> futureResponse =
           getGeoPlacesClient().reverseGeocode(request);

       return futureResponse.whenComplete((response, exception) -> {
           if (exception != null) {
               Throwable cause = exception.getCause();
               if (cause instanceof
software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                   throw new CompletionException("A validation error occurred: " +
cause.getMessage(), cause);
               }
               throw new CompletionException("Error performing reverse geocoding",
exception);
           }

           response.resultItems().forEach(result ->
               logger.info("The address is: " + result.address().label())
           );
       });
   }


   /**
    * Calculates the distance between two locations asynchronously.
    *
    * @param routeCalcName the name of the route calculator to use
    * @return a {@link CompletableFuture} that will complete with a {@link
CalculateRouteResponse} containing the distance and estimated duration of the route
    */
```

```
   public CompletableFuture<CalculateRouteResponse> calcDistanceAsync(String
routeCalcName) {
       // Define coordinates for Seattle, WA and Vancouver, BC.
       List<Double> departurePosition = Arrays.asList(-122.3321, 47.6062);
       List<Double> arrivePosition = Arrays.asList(-123.1216, 49.2827);

       CalculateRouteRequest request = CalculateRouteRequest.builder()
           .calculatorName(routeCalcName)
           .departurePosition(departurePosition)
           .destinationPosition(arrivePosition)
           .travelMode("Car") // Options: Car, Truck, Walking, Bicycle
           .distanceUnit("Kilometers") // Options: Meters, Kilometers, Miles
           .build();

       return getClient().calculateRoute(request)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The AWS resource was not
found: " + cause.getMessage(), cause);
                   }
                   throw new CompletionException("Failed to calculate route: " +
exception.getMessage(), exception);
               }
           });
   }


   /**
    * Creates a new route calculator with the specified name and data source.
    *
    * @param routeCalcName the name of the route calculator to be created
    */
   public CompletableFuture<CreateRouteCalculatorResponse>
createRouteCalculator(String routeCalcName) {
       String dataSource = "Esri"; // or "Here"
       CreateRouteCalculatorRequest request =
CreateRouteCalculatorRequest.builder()
           .calculatorName(routeCalcName)
           .dataSource(dataSource)
           .build();

       return getClient().createRouteCalculator(request)
```

```
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflict error occurred: "
+ cause.getMessage(), cause);
                    }
                    throw new CompletionException("Failed to create route
calculator: " + exception.getMessage(), exception);
                }
            });
    }


    /**
     * Retrieves the position of a device using the provided LocationClient.
     *
     * @param trackerName The name of the tracker associated with the device.
     * @param deviceId    The ID of the device to retrieve the position for.
     * @throws RuntimeException If there is an error fetching the device position.
     */
    public CompletableFuture<GetDevicePositionResponse> getDevicePosition(String
trackerName, String deviceId) {
        GetDevicePositionRequest request = GetDevicePositionRequest.builder()
            .trackerName(trackerName)
            .deviceId(deviceId)
            .build();

        return getClient().getDevicePosition(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The AWS resource was not
found: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error fetching device position: "
+ exception.getMessage(), exception);
                }
            });
    }


    /**
```

```
    * Updates the position of a device in the location tracking system.
    *
    * @param trackerName the name of the tracker associated with the device
    * @param deviceId    the unique identifier of the device
    * @throws RuntimeException if an error occurs while updating the device
position
    */
   public CompletableFuture<BatchUpdateDevicePositionResponse>
updateDevicePosition(String trackerName, String deviceId) {
       double latitude = 37.7749;  // Example: San Francisco
       double longitude = -122.4194;

       DevicePositionUpdate positionUpdate = DevicePositionUpdate.builder()
           .deviceId(deviceId)
           .sampleTime(Instant.now()) // Timestamp of position update.
           .position(Arrays.asList(longitude, latitude)) // AWS requires
[longitude, latitude]
           .build();

       BatchUpdateDevicePositionRequest request =
BatchUpdateDevicePositionRequest.builder()
           .trackerName(trackerName)
           .updates(positionUpdate)
           .build();

       CompletableFuture<BatchUpdateDevicePositionResponse> futureResponse =
getClient().batchUpdateDevicePosition(request);
       return futureResponse.whenComplete((response, exception) -> {
           if (exception != null) {
               Throwable cause = exception.getCause();
               if (cause instanceof ResourceNotFoundException) {
                   throw new CompletionException("The resource was not found: " +
cause.getMessage(), cause);
               } else {
                   throw new CompletionException("Error updating device position: "
+ exception.getMessage(), exception);
               }
           }
       });
   }


   /**
```

```
    * Creates a new tracker resource in your AWS account, which you can use to
track the location of devices.
    *
    * @param trackerName the name of the tracker to be created
    * @return a {@link CompletableFuture} that, when completed, will contain the
Amazon Resource Name (ARN) of the created tracker
    */
   public CompletableFuture<String> createTracker(String trackerName) {
       CreateTrackerRequest trackerRequest = CreateTrackerRequest.builder()
           .description("Created using the Java V2 SDK")
           .trackerName(trackerName)
           .positionFiltering("TimeBased") // Options: TimeBased, DistanceBased,
AccuracyBased
           .build();

       return getClient().createTracker(trackerRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ConflictException) {
                       throw new CompletionException("Conflict occurred while
creating tracker: " + cause.getMessage(), cause);
                   }
                   throw new CompletionException("Error creating tracker: " +
exception.getMessage(), exception);
               }
           })
           .thenApply(CreateTrackerResponse::trackerArn); // Return only the
tracker ARN
   }



   /**
    * Adds a new geofence to the specified collection.
    *
    * @param collectionName the name of the geofence collection to add the geofence
to
    * @param geoId          the unique identifier for the geofence
    */
   public CompletableFuture<PutGeofenceResponse> putGeofence(String collectionName,
String geoId) {
       // Define the geofence geometry (polygon).
       GeofenceGeometry geofenceGeometry = GeofenceGeometry.builder()
```

```
            .polygon(List.of(
                List.of(
                    List.of(-122.3381, 47.6101), // First point
                    List.of(-122.3281, 47.6101),
                    List.of(-122.3281, 47.6201),
                    List.of(-122.3381, 47.6201),
                    List.of(-122.3381, 47.6101) // Closing the polygon
                )
            ))
            .build();

        PutGeofenceRequest geofenceRequest = PutGeofenceRequest.builder()
            .collectionName(collectionName) // Specify the collection.
            .geofenceId(geoId) // Unique ID for the geofence.
            .geometry(geofenceGeometry)
            .build();

        return getClient().putGeofence(geofenceRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ValidationException) {
                        throw new CompletionException("Validation error while
creating geofence: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error creating geofence: " +
exception.getMessage(), exception);
                }
            });
    }


    /**
     * Creates a new geofence collection.
     *
     * @param collectionName the name of the geofence collection to be created
     */
    public CompletableFuture<String> createGeofenceCollection(String collectionName)
{
        CreateGeofenceCollectionRequest collectionRequest =
CreateGeofenceCollectionRequest.builder()
            .collectionName(collectionName)
            .description("Created by using the AWS SDK for Java")
            .build();
```

```
        return getClient().createGeofenceCollection(collectionRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("The geofence collection was
not created due to ConflictException.", cause);
                    }
                    throw new CompletionException("Failed to create geofence
collection: " + exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.collectionArn()); // Return only the ARN
    }




    /**
     * Creates a new API key with the specified name and restrictions.
     *
     * @param keyName the name of the API key to be created
     * @param mapArn  the Amazon Resource Name (ARN) of the map resource to which
the API key will be associated
     * @return a {@link CompletableFuture} that completes with the Amazon Resource
Name (ARN) of the created API key,
     * or {@code null} if the operation failed
     */
    public CompletableFuture<String> createKey(String keyName, String mapArn) {
        ApiKeyRestrictions keyRestrictions = ApiKeyRestrictions.builder()
            .allowActions("geo:GetMap*")
            .allowResources(mapArn)
            .build();

        CreateKeyRequest request = CreateKeyRequest.builder()
            .keyName(keyName)
            .restrictions(keyRestrictions)
            .noExpiry(true)
            .build();

        return getClient().createKey(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
```

```
                    Throwable cause = exception.getCause();
                    if (cause instanceof AccessDeniedException) {
                        throw new CompletionException("The request was denied
because of insufficient access or permissions.", cause);
                    }
                    throw new CompletionException("Failed to create API key: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.keyArn()); // This will never return
null if the response reaches here
    }



    /**
     * Creates a new map with the specified name and configuration.
     *
     * @param mapName the name of the map to be created
     * @return a {@link CompletableFuture} that, when completed, will contain the
Amazon Resource Name (ARN) of the created map
     * @throws CompletionException if an error occurs while creating the map, such
as exceeding the service quota
     */
    public CompletableFuture<String> createMap(String mapName) {
        MapConfiguration configuration = MapConfiguration.builder()
            .style("VectorEsriNavigation")
            .build();

        CreateMapRequest mapRequest = CreateMapRequest.builder()
            .mapName(mapName)
            .configuration(configuration)
            .description("A map created using the Java V2 API")
            .build();

        return getClient().createMap(mapRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ServiceQuotaExceededException) {
                        throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                    }
```

```
                    throw new CompletionException("Failed to create map: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.mapArn()); // Return the map ARN
    }



    /**
     * Deletes a geofence collection asynchronously.
     *
     * @param collectionName the name of the geofence collection to be deleted
     * @return a {@link CompletableFuture} that completes when the geofence
collection has been deleted
     */
    public CompletableFuture<Void> deleteGeofenceCollectionAsync(String
collectionName) {
        DeleteGeofenceCollectionRequest collectionRequest =
DeleteGeofenceCollectionRequest.builder()
            .collectionName(collectionName)
            .build();

        return getClient().deleteGeofenceCollection(collectionRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested geofence
collection was not found.", cause);
                    }
                    throw new CompletionException("Failed to delete geofence
collection: " + exception.getMessage(), exception);
                }
                logger.info("The geofence collection {} was deleted.",
collectionName);
            })
            .thenApply(response -> null);
    }



    /**
     * Deletes the specified key from the key-value store.
```

```
     *
     * @param keyName the name of the key to be deleted
     * @return a {@link CompletableFuture} that completes when the key has been
deleted
     * @throws CompletionException if the key was not found or if an error occurred
during the deletion process
     */
    public CompletableFuture<Void> deleteKey(String keyName) {
        DeleteKeyRequest keyRequest = DeleteKeyRequest.builder()
            .keyName(keyName)
            .build();

        return getClient().deleteKey(keyRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The key was not found.",
cause);
                    }
                    throw new CompletionException("Failed to delete key: " +
exception.getMessage(), exception);
                }
                logger.info("The key {} was deleted.", keyName);
            })
            .thenApply(response -> null);
    }


    /**
     * Deletes a map with the specified name.
     *
     * @param mapName the name of the map to be deleted
     * @return a {@link CompletableFuture} that completes when the map deletion is
successful, or throws a {@link CompletionException} if an error occurs
     */
    public CompletableFuture<Void> deleteMap(String mapName) {
        DeleteMapRequest mapRequest = DeleteMapRequest.builder()
            .mapName(mapName)
            .build();

        return getClient().deleteMap(mapRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
```

```
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The map was not found.",
cause);
                    }
                    throw new CompletionException("Failed to delete map: " +
exception.getMessage(), exception);
                }
                logger.info("The map {} was deleted.", mapName);
            })
            .thenApply(response -> null);
    }


    /**
     * Deletes a tracker with the specified name.
     *
     * @param trackerName the name of the tracker to be deleted
     * @return a {@link CompletableFuture} that completes when the tracker has been
deleted
     * @throws CompletionException if an error occurs while deleting the tracker
     *                             - if the tracker was not found, a {@link
ResourceNotFoundException} is thrown wrapped in the CompletionException
     *                             - if any other error occurs, a generic
CompletionException is thrown with the error message
     */
    public CompletableFuture<Void> deleteTracker(String trackerName) {
        DeleteTrackerRequest trackerRequest = DeleteTrackerRequest.builder()
            .trackerName(trackerName)
            .build();

        return getClient().deleteTracker(trackerRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The tracker was not found.",
cause);
                    }
                    throw new CompletionException("Failed to delete the tracker: " +
exception.getMessage(), exception);
                }
                logger.info("The tracker {} was deleted.", trackerName);
            })
```

```
            .thenApply(response -> null); // Ensures CompletableFuture<Void>
    }


    /**
     * Deletes a route calculator from the system.
     *
     * @param calcName the name of the route calculator to delete
     * @return a {@link CompletableFuture} that completes when the route calculator
 has been deleted
     * @throws CompletionException if an error occurs while deleting the route
 calculator
     *                             - If the route calculator was not found, a {@link
 ResourceNotFoundException} will be thrown
     *                             - If any other error occurs, a generic {@link
 CompletionException} will be thrown
     */
    public CompletableFuture<Void> deleteRouteCalculator(String calcName) {
        DeleteRouteCalculatorRequest calculatorRequest =
 DeleteRouteCalculatorRequest.builder()
            .calculatorName(calcName)
            .build();

        return getClient().deleteRouteCalculator(calculatorRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The route calculator was not
 found.", cause);
                    }
                    throw new CompletionException("Failed to delete the route
 calculator: " + exception.getMessage(), exception);
                }
                logger.info("The route calculator {} was deleted.", calcName);
            })
            .thenApply(response -> null);
    }
}
```

操作

`BatchUpdateDevicePosition`

以下代码示例演示了如何使用 `BatchUpdateDevicePosition`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Updates the position of a device in the location tracking system.
    *
    * @param trackerName the name of the tracker associated with the device
    * @param deviceId    the unique identifier of the device
    * @throws RuntimeException if an error occurs while updating the device
position
    */
   public CompletableFuture<BatchUpdateDevicePositionResponse>
updateDevicePosition(String trackerName, String deviceId) {
       double latitude = 37.7749;  // Example: San Francisco
       double longitude = -122.4194;

       DevicePositionUpdate positionUpdate = DevicePositionUpdate.builder()
           .deviceId(deviceId)
           .sampleTime(Instant.now()) // Timestamp of position update.
           .position(Arrays.asList(longitude, latitude)) // AWS requires
[longitude, latitude]
           .build();

       BatchUpdateDevicePositionRequest request =
BatchUpdateDevicePositionRequest.builder()
           .trackerName(trackerName)
           .updates(positionUpdate)
           .build();
```

```
        CompletableFuture<BatchUpdateDevicePositionResponse> futureResponse =
getClient().batchUpdateDevicePosition(request);
        return futureResponse.whenComplete((response, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    throw new CompletionException("The resource was not found: " +
cause.getMessage(), cause);
                } else {
                    throw new CompletionException("Error updating device position: "
+ exception.getMessage(), exception);
                }
            }
        });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[BatchUpdateDevicePosition](#)中的。

`CalculateRoute`

以下代码示例演示了如何使用 `CalculateRoute`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Calculates the distance between two locations asynchronously.
     *
     * @param routeCalcName the name of the route calculator to use
     * @return a {@link CompletableFuture} that will complete with a {@link
CalculateRouteResponse} containing the distance and estimated duration of the route
     */
```

```
    public CompletableFuture<CalculateRouteResponse> calcDistanceAsync(String
routeCalcName) {
        // Define coordinates for Seattle, WA and Vancouver, BC.
        List<Double> departurePosition = Arrays.asList(-122.3321, 47.6062);
        List<Double> arrivePosition = Arrays.asList(-123.1216, 49.2827);

        CalculateRouteRequest request = CalculateRouteRequest.builder()
            .calculatorName(routeCalcName)
            .departurePosition(departurePosition)
            .destinationPosition(arrivePosition)
            .travelMode("Car") // Options: Car, Truck, Walking, Bicycle
            .distanceUnit("Kilometers") // Options: Meters, Kilometers, Miles
            .build();

        return getClient().calculateRoute(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The AWS resource was not
found: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Failed to calculate route: " +
exception.getMessage(), exception);
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CalculateRoute中的。

`CreateGeofenceCollection`

以下代码示例演示了如何使用 `CreateGeofenceCollection`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Creates a new geofence collection.
    *
    * @param collectionName the name of the geofence collection to be created
    */
   public CompletableFuture<String> createGeofenceCollection(String collectionName)
{
       CreateGeofenceCollectionRequest collectionRequest =
CreateGeofenceCollectionRequest.builder()
           .collectionName(collectionName)
           .description("Created by using the AWS SDK for Java")
           .build();

       return getClient().createGeofenceCollection(collectionRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ConflictException) {
                       throw new CompletionException("The geofence collection was
not created due to ConflictException.", cause);
                   }
                   throw new CompletionException("Failed to create geofence
collection: " + exception.getMessage(), exception);
               }
           })
           .thenApply(response -> response.collectionArn()); // Return only the ARN
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateGeofenceCollection中的。

CreateKey

以下代码示例演示了如何使用 CreateKey。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Creates a new API key with the specified name and restrictions.
    *
    * @param keyName the name of the API key to be created
    * @param mapArn  the Amazon Resource Name (ARN) of the map resource to which
the API key will be associated
    * @return a {@link CompletableFuture} that completes with the Amazon Resource
Name (ARN) of the created API key,
    * or {@code null} if the operation failed
    */
   public CompletableFuture<String> createKey(String keyName, String mapArn) {
       ApiKeyRestrictions keyRestrictions = ApiKeyRestrictions.builder()
           .allowActions("geo:GetMap*")
           .allowResources(mapArn)
           .build();

       CreateKeyRequest request = CreateKeyRequest.builder()
           .keyName(keyName)
           .restrictions(keyRestrictions)
           .noExpiry(true)
           .build();

       return getClient().createKey(request)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof AccessDeniedException) {
                       throw new CompletionException("The request was denied
because of insufficient access or permissions.", cause);
                   }
                   throw new CompletionException("Failed to create API key: " +
exception.getMessage(), exception);
               }
```

```
            })
            .thenApply(response -> response.keyArn()); // This will never return
null if the response reaches here
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateKey中的。

`CreateMap`

以下代码示例演示了如何使用 `CreateMap`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Creates a new map with the specified name and configuration.
     *
     * @param mapName the name of the map to be created
     * @return a {@link CompletableFuture} that, when completed, will contain the
Amazon Resource Name (ARN) of the created map
     * @throws CompletionException if an error occurs while creating the map, such
as exceeding the service quota
     */
    public CompletableFuture<String> createMap(String mapName) {
        MapConfiguration configuration = MapConfiguration.builder()
            .style("VectorEsriNavigation")
            .build();

        CreateMapRequest mapRequest = CreateMapRequest.builder()
            .mapName(mapName)
            .configuration(configuration)
            .description("A map created using the Java V2 API")
            .build();
```

```
        return getClient().createMap(mapRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ServiceQuotaExceededException) {
                        throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                    }
                    throw new CompletionException("Failed to create map: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(response -> response.mapArn()); // Return the map ARN
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateMap中的。

`CreateRouteCalculator`

以下代码示例演示了如何使用 `CreateRouteCalculator`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Creates a new route calculator with the specified name and data source.
     *
     * @param routeCalcName the name of the route calculator to be created
     */
    public CompletableFuture<CreateRouteCalculatorResponse>
createRouteCalculator(String routeCalcName) {
        String dataSource = "Esri"; // or "Here"
        CreateRouteCalculatorRequest request =
CreateRouteCalculatorRequest.builder()
```

```
            .calculatorName(routeCalcName)
            .dataSource(dataSource)
            .build();

        return getClient().createRouteCalculator(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("A conflict error occurred: "
+ cause.getMessage(), cause);
                    }
                    throw new CompletionException("Failed to create route
calculator: " + exception.getMessage(), exception);
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateRouteCalculator中的。

`CreateTracker`

以下代码示例演示了如何使用 `CreateTracker`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Creates a new tracker resource in your AWS account, which you can use to
track the location of devices.
     *
     * @param trackerName the name of the tracker to be created
     * @return a {@link CompletableFuture} that, when completed, will contain the
Amazon Resource Name (ARN) of the created tracker
     */
```

```
    public CompletableFuture<String> createTracker(String trackerName) {
        CreateTrackerRequest trackerRequest = CreateTrackerRequest.builder()
            .description("Created using the Java V2 SDK")
            .trackerName(trackerName)
            .positionFiltering("TimeBased") // Options: TimeBased, DistanceBased,
AccuracyBased
            .build();

        return getClient().createTracker(trackerRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ConflictException) {
                        throw new CompletionException("Conflict occurred while
creating tracker: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error creating tracker: " +
exception.getMessage(), exception);
                }
            })
            .thenApply(CreateTrackerResponse::trackerArn); // Return only the
tracker ARN
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateTracker中的。

`DeleteGeofenceCollection`

以下代码示例演示了如何使用 `DeleteGeofenceCollection`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Deletes a geofence collection asynchronously.
     *
     * @param collectionName the name of the geofence collection to be deleted
     * @return a {@link CompletableFuture} that completes when the geofence
collection has been deleted
     */
    public CompletableFuture<Void> deleteGeofenceCollectionAsync(String
collectionName) {
        DeleteGeofenceCollectionRequest collectionRequest =
DeleteGeofenceCollectionRequest.builder()
            .collectionName(collectionName)
            .build();

        return getClient().deleteGeofenceCollection(collectionRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The requested geofence
collection was not found.", cause);
                    }
                    throw new CompletionException("Failed to delete geofence
collection: " + exception.getMessage(), exception);
                }
                logger.info("The geofence collection {} was deleted.",
collectionName);
            })
            .thenApply(response -> null);
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteGeofenceCollection中的。

DeleteKey

以下代码示例演示了如何使用 DeleteKey。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Deletes the specified key from the key-value store.
    *
    * @param keyName the name of the key to be deleted
    * @return a {@link CompletableFuture} that completes when the key has been
deleted
    * @throws CompletionException if the key was not found or if an error occurred
during the deletion process
    */
   public CompletableFuture<Void> deleteKey(String keyName) {
       DeleteKeyRequest keyRequest = DeleteKeyRequest.builder()
           .keyName(keyName)
           .build();

       return getClient().deleteKey(keyRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The key was not found.",
cause);
                   }
                   throw new CompletionException("Failed to delete key: " +
exception.getMessage(), exception);
               }
               logger.info("The key {} was deleted.", keyName);
           })
           .thenApply(response -> null);
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteKey中的。

DeleteMap

以下代码示例演示了如何使用 DeleteMap。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Deletes a map with the specified name.
     *
     * @param mapName the name of the map to be deleted
     * @return a {@link CompletableFuture} that completes when the map deletion is
successful, or throws a {@link CompletionException} if an error occurs
     */
    public CompletableFuture<Void> deleteMap(String mapName) {
        DeleteMapRequest mapRequest = DeleteMapRequest.builder()
            .mapName(mapName)
            .build();

        return getClient().deleteMap(mapRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The map was not found.",
cause);
                    }
                    throw new CompletionException("Failed to delete map: " +
exception.getMessage(), exception);
                }
                logger.info("The map {} was deleted.", mapName);
            })
            .thenApply(response -> null);
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteMap中的。

`DeleteRouteCalculator`

以下代码示例演示了如何使用 `DeleteRouteCalculator`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Deletes a route calculator from the system.
    *
    * @param calcName the name of the route calculator to delete
    * @return a {@link CompletableFuture} that completes when the route calculator
has been deleted
    * @throws CompletionException if an error occurs while deleting the route
calculator
    *                             - If the route calculator was not found, a {@link
ResourceNotFoundException} will be thrown
    *                             - If any other error occurs, a generic {@link
CompletionException} will be thrown
    */
   public CompletableFuture<Void> deleteRouteCalculator(String calcName) {
       DeleteRouteCalculatorRequest calculatorRequest =
DeleteRouteCalculatorRequest.builder()
           .calculatorName(calcName)
           .build();

       return getClient().deleteRouteCalculator(calculatorRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   Throwable cause = exception.getCause();
                   if (cause instanceof ResourceNotFoundException) {
                       throw new CompletionException("The route calculator was not
found.", cause);
                   }
```

```
                    throw new CompletionException("Failed to delete the route
calculator: " + exception.getMessage(), exception);
                }
                logger.info("The route calculator {} was deleted.", calcName);
            })
            .thenApply(response -> null);
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteRouteCalculator中的。

`DeleteTracker`

以下代码示例演示了如何使用 `DeleteTracker`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Deletes a tracker with the specified name.
     *
     * @param trackerName the name of the tracker to be deleted
     * @return a {@link CompletableFuture} that completes when the tracker has been
deleted
     * @throws CompletionException if an error occurs while deleting the tracker
     *                             - if the tracker was not found, a {@link
ResourceNotFoundException} is thrown wrapped in the CompletionException
     *                             - if any other error occurs, a generic
CompletionException is thrown with the error message
     */
    public CompletableFuture<Void> deleteTracker(String trackerName) {
        DeleteTrackerRequest trackerRequest = DeleteTrackerRequest.builder()
            .trackerName(trackerName)
            .build();

        return getClient().deleteTracker(trackerRequest)
```

```
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The tracker was not found.",
cause);
                    }
                    throw new CompletionException("Failed to delete the tracker: " +
exception.getMessage(), exception);
                }
                logger.info("The tracker {} was deleted.", trackerName);
            })
            .thenApply(response -> null); // Ensures CompletableFuture<Void>
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteTracker中的。

`GetDevicePosition`

以下代码示例演示了如何使用 `GetDevicePosition`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Retrieves the position of a device using the provided LocationClient.
     *
     * @param trackerName The name of the tracker associated with the device.
     * @param deviceId    The ID of the device to retrieve the position for.
     * @throws RuntimeException If there is an error fetching the device position.
     */
    public CompletableFuture<GetDevicePositionResponse> getDevicePosition(String
trackerName, String deviceId) {
        GetDevicePositionRequest request = GetDevicePositionRequest.builder()
```

```
            .trackerName(trackerName)
            .deviceId(deviceId)
            .build();

        return getClient().getDevicePosition(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ResourceNotFoundException) {
                        throw new CompletionException("The AWS resource was not
found: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error fetching device position: "
+ exception.getMessage(), exception);
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[GetDevicePosition](#)中的。

`PutGeofence`

以下代码示例演示了如何使用 `PutGeofence`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Adds a new geofence to the specified collection.
     *
     * @param collectionName the name of the geofence collection to add the geofence
to
     * @param geoId          the unique identifier for the geofence
     */
```

```
    public CompletableFuture<PutGeofenceResponse> putGeofence(String collectionName,
String geoId) {
        // Define the geofence geometry (polygon).
        GeofenceGeometry geofenceGeometry = GeofenceGeometry.builder()
            .polygon(List.of(
                List.of(
                    List.of(-122.3381, 47.6101), // First point
                    List.of(-122.3281, 47.6101),
                    List.of(-122.3281, 47.6201),
                    List.of(-122.3381, 47.6201),
                    List.of(-122.3381, 47.6101) // Closing the polygon
                )
            ))
            .build();

        PutGeofenceRequest geofenceRequest = PutGeofenceRequest.builder()
            .collectionName(collectionName) // Specify the collection.
            .geofenceId(geoId) // Unique ID for the geofence.
            .geometry(geofenceGeometry)
            .build();

        return getClient().putGeofence(geofenceRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof ValidationException) {
                        throw new CompletionException("Validation error while
creating geofence: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error creating geofence: " +
exception.getMessage(), exception);
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutGeofence中的。

使用适用于 Java 的 SDK 2.x 的 Location Service Places 示例

以下代码示例向您展示了如何使用 with Location Service Places 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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从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

`ReverseGeocode`

以下代码示例演示了如何使用 `ReverseGeocode`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Performs reverse geocoding using the AWS Geo Places API.
    * Reverse geocoding is the process of converting geographic coordinates
(latitude and longitude) to a human-readable address.
    * This method uses the latitude and longitude of San Francisco as the input,
and prints the resulting address.
    */
   public CompletableFuture<ReverseGeocodeResponse> reverseGeocode() {
       double latitude = 37.7749;  // San Francisco
       double longitude = -122.4194;
       logger.info("Use latitude 37.7749 and longitude -122.4194");

       // AWS expects [longitude, latitude].
       List<Double> queryPosition = List.of(longitude, latitude);
       ReverseGeocodeRequest request = ReverseGeocodeRequest.builder()
           .queryPosition(queryPosition)
           .build();
```

```
        CompletableFuture<ReverseGeocodeResponse> futureResponse =
            getGeoPlacesClient().reverseGeocode(request);

        return futureResponse.whenComplete((response, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof
 software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                    throw new CompletionException("A validation error occurred: " +
 cause.getMessage(), cause);
                }
                throw new CompletionException("Error performing reverse geocoding",
 exception);
            }

            response.resultItems().forEach(result ->
                logger.info("The address is: " + result.address().label())
            );
        });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ReverseGeocode中的。

`SearchNearby`

以下代码示例演示了如何使用 `SearchNearby`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Performs a nearby places search based on the provided geographic coordinates
 (latitude and longitude).
     * The method sends an asynchronous request to search for places within a 1-
kilometer radius of the specified location.
```

```
     * The results are processed and printed once the search completes successfully.
     */
    public CompletableFuture<SearchNearbyResponse> searchNearBy() {
        double latitude = 37.7749;  // San Francisco
        double longitude = -122.4194;
        List<Double> queryPosition = List.of(longitude, latitude);

        // Set up the request for searching nearby places.
        SearchNearbyRequest request = SearchNearbyRequest.builder()
            .queryPosition(queryPosition)  // Set the position
            .queryRadius(1000L)  // Radius in meters (1000 meters = 1 km).
            .build();

        return getGeoPlacesClient().searchNearby(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof
software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                        throw new CompletionException("A validation error occurred:
" + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error performing place search",
exception);
                }

                // Process the response and print the results.
                response.resultItems().forEach(result -> {
                    logger.info("Place Name: " + result.placeType().name());
                    logger.info("Address: " + result.address().label());
                    logger.info("Distance: " + result.distance() + " meters");
                    logger.info("-------------------------");
                });
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SearchNearby中的。

SearchText

以下代码示例演示了如何使用 SearchText。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Searches for a place using the provided search query and prints the detailed
information of the first result.
    *
    * @param searchQuery the search query to be used for the place search (ex,
coffee shop)
    */
   public CompletableFuture<Void> searchText(String searchQuery) {
       double latitude = 37.7749;  // San Francisco
       double longitude = -122.4194;
       List<Double> queryPosition = List.of(longitude, latitude);

       SearchTextRequest request = SearchTextRequest.builder()
               .queryText(searchQuery)
               .biasPosition(queryPosition)
               .build();

       return getGeoPlacesClient().searchText(request)
               .thenCompose(response -> {
                   if (response.resultItems().isEmpty()) {
                       logger.info("No places found.");
                       return CompletableFuture.completedFuture(null);
                   }

                   // Get the first place ID
                   String placeId = response.resultItems().get(0).placeId();
                   logger.info("Found Place with id: " + placeId);

                   // Fetch detailed info using getPlace
                   GetPlaceRequest getPlaceRequest = GetPlaceRequest.builder()
                           .placeId(placeId)
                           .build();
```

```
                    return getGeoPlacesClient().getPlace(getPlaceRequest)
                            .thenAccept(placeResponse -> {
                                logger.info("Detailed Place Information:");
                                logger.info("Name: " +
 placeResponse.placeType().name());
                                logger.info("Address: " +
 placeResponse.address().label());

                                if (placeResponse.foodTypes() != null && !
placeResponse.foodTypes().isEmpty()) {
                                    logger.info("Food Types:");
                                    placeResponse.foodTypes().forEach(foodType -> {
                                        logger.info("  - " + foodType);
                                    });
                                } else {
                                    logger.info("No food types available.");
                                }
                                logger.info("-------------------------");
                            });
                })
                .exceptionally(exception -> {
                    Throwable cause = exception.getCause();
                    if (cause instanceof
 software.amazon.awssdk.services.geoplaces.model.ValidationException) {
                        throw new CompletionException("A validation error occurred:
 " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Error performing place search",
 exception);
                });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SearchText中的。

AWS Marketplace 使用适用于 Java 的 SDK 2.x 编目 API 示例

以下代码示例向您展示了如何使用 with Catalo AWS Marketplace g API 执行操作和实现常见场景。
AWS SDK for Java 2.x

每个示例都包含一个指向完整源代码的链接，您可以从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
- 优惠
- 产品
- 转售授权
- SaaS 产品
- 公用事业机构

AMI 产品

向现有 AMI 产品添加维度并更新优惠定价条款

以下代码示例演示如何向现有 AMI 产品添加维度并更新优惠定价条款。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "AddDimensions",
            "Entity": {
```

```
            "Identifier": "prod-1111111111111",
            "Type": "AmiProduct@1.0"
        },
        "DetailsDocument": [
            {
                "Key": "m7g.8xlarge",
                "Description": "m7g.8xlarge",
                "Name": "m7g.8xlarge",
                "Types": [
                    "Metered"
                ],
                "Unit": "Hrs"
            }
        ]
    },
    {
        "ChangeType": "UpdatePricingTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "offer-1111111111111"
        },
        "DetailsDocument": {
            "PricingModel": "Usage",
            "Terms": [
                {
                    "Type": "UsageBasedPricingTerm",
                    "CurrencyCode": "USD",
                    "RateCards": [
                        {
                            "RateCard": [
                                {
                                    "DimensionKey": "m5.large",
                                    "Price": "0.15"
                                },
                                {
                                    "DimensionKey": "m7g.4xlarge",
                                    "Price": "0.45"
                                },
                                {
                                    "DimensionKey": "m7g.2xlarge",
                                    "Price": "0.45"
                                },
                                {
                                    "DimensionKey": "m7g.8xlarge",
```

```
                                    "Price": "0.55"
                                }
                            ]
                        }
                    ]
                }
            ]
        }
    ]
}
```

- 有关 API 的详细信息，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

以下代码示例演示如何添加部署 AMI 产品的区域。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-2",
                    "us-west-2"
```

```
                ]
            }
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

创建公开或受限 AMI 产品和按小时按年定价的公开优惠

以下代码示例演示如何创建公开或受限 AMI 产品和按小时按年定价的公开优惠。此示例创建标准或自定义 EULA。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "InstanceTypes": [
                    "t2.micro"
                ]
            }
```

```
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
                "DeliveryOptions": [
                    {
                        "Details": {
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-1111111111111111",
                                    "AccessRoleArn":
"arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
                                "SecurityGroups": [
                                    {
                                        "IpProtocol": "tcp",
                                        "IpRanges": [
                                            "0.0.0.0/0"
                                        ],
                                        "FromPort": 10,
                                        "ToPort": 22
                                    }
                                ]
                            }
                        }
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "t2.micro",
                    "Description": "t2.micro",
                    "Name": "t2.micro",
                    "Types": [
                        "Metered"
                    ],
                    "Unit": "Hrs"
                }
            ]
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
```

```
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with hourly-annual pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.15"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
```

```
                    "CurrencyCode": "USD",
                    "RateCards": [
                        {
                            "Selector": {
                                "Type": "Duration",
                                "Value": "P365D"
                            },
                            "RateCard": [
                                {
                                    "DimensionKey": "t2.micro",
                                    "Price": "150"
                                }
                            ],
                            "Constraints": {
                                "MultipleDimensionSelection": "Allowed",
                                "QuantityConfiguration": "Allowed"
                            }
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateLegalTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "LegalTerm",
                    "Documents": [
                        {
                            "Type": "StandardEula",
                            "Version": "2022-07-14"
                        }
                    ]
                }
            ]
        }
    },
    {
```

```
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

创建公开或受限 AMI 产品和按小时按月定价的公开优惠

以下代码示例演示如何创建公开或受限 AMI 产品和按小时按月定价的公开优惠。此示例创建标准或自定义 EULA。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 RunChangesets。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "InstanceTypes": [
                    "t2.micro"
                ]
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
                "DeliveryOptions": [
                    {
                        "Details": {
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-11111111111111111",
                                    "AccessRoleArn":
"arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
                                },
                                "UsageInstructions": "Test AMI Version",
                                "RecommendedInstanceType": "t2.micro",
```

```
                            "SecurityGroups": [
                                {
                                    "IpProtocol": "tcp",
                                    "IpRanges": [
                                        "0.0.0.0/0"
                                    ],
                                    "FromPort": 10,
                                    "ToPort": 22
                                }
                            ]
                        }
                    }
                }
            ]
        }
    },
    {
        "ChangeType": "AddDimensions",
        "Entity": {
            "Type": "AmiProduct@1.0",
            "Identifier": "$CreateProductChange.Entity.Identifier"
        },
        "DetailsDocument": [
            {
                "Key": "t2.micro",
                "Description": "t2.micro",
                "Name": "t2.micro",
                "Types": [
                    "Metered"
                ],
                "Unit": "Hrs"
            }
        ]
    },
    {
        "ChangeType": "UpdateTargeting",
        "Entity": {
            "Type": "AmiProduct@1.0",
            "Identifier": "$CreateProductChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "PositiveTargeting": {
                "BuyerAccounts": [
                    "111111111111",
```

```
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with hourly-monthly pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

创建公开或受限 AMI 产品和按小时定价的公开优惠

以下代码示例演示如何创建公开或受限 AMI 产品和按小时定价的公开优惠。此示例创建标准或自定义 EULA。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 RunChangesets。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "AmiProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Operating Systems"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddRegions",
            "Entity": {
```

```
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-east-1"
                ]
            }
        },
        {
            "ChangeType": "AddInstanceTypes",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "InstanceTypes": [
                    "t2.micro"
                ]
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Version": {
                    "VersionTitle": "Test AMI Version1.0",
                    "ReleaseNotes": "Test AMI Version"
                },
                "DeliveryOptions": [
                    {
                        "Details": {
                            "AmiDeliveryOptionDetails": {
                                "AmiSource": {
                                    "AmiId": "ami-1111111111111111",
                                    "AccessRoleArn":
"arn:aws:iam::111111111111:role/AWSMarketplaceAmiIngestion",
                                    "UserName": "ec2-user",
                                    "OperatingSystemName": "AMAZONLINUX",
                                    "OperatingSystemVersion": "10.0.14393",
                                    "ScanningPort": 22
```

```
                            },
                            "UsageInstructions": "Test AMI Version",
                            "RecommendedInstanceType": "t2.micro",
                            "SecurityGroups": [
                                {
                                    "IpProtocol": "tcp",
                                    "IpRanges": [
                                        "0.0.0.0/0"
                                    ],
                                    "FromPort": 10,
                                    "ToPort": 22
                                }
                            ]
                        }
                    }
                }
            ]
        }
    },
    {
        "ChangeType": "AddDimensions",
        "Entity": {
            "Type": "AmiProduct@1.0",
            "Identifier": "$CreateProductChange.Entity.Identifier"
        },
        "DetailsDocument": [
            {
                "Key": "t2.micro",
                "Description": "t2.micro",
                "Name": "t2.micro",
                "Types": [
                    "Metered"
                ],
                "Unit": "Hrs"
            }
        ]
    },
    {
        "ChangeType": "UpdateTargeting",
        "Entity": {
            "Type": "AmiProduct@1.0",
            "Identifier": "$CreateProductChange.Entity.Identifier"
        },
        "DetailsDocument": {
```

```
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for AmiProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with hourly pricing for AmiProduct
using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

创建采用公开优惠草稿的 AMI 产品草稿

以下代码示例演示如何创建采用公开优惠草稿的 AMI 产品草稿。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
```

```
    "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 的详细信息，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

以下代码示例演示如何限制部署 AMI 产品的区域。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 RunChangesets。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "RestrictRegions",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "Regions": [
                    "us-west-2"
                ]
            }
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

限制产品可见性

以下代码示例演示如何限制产品可见性。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 RunChangesets。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateVisibility",
```

```
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "TargetVisibility": "Restricted"
            }
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

指定是否在新区域中部署 AMI 资产

以下代码示例说明如何指定 AMI 资产是否部署在为支持 future 区域 AWS 而构建的新区域。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 RunChangesets。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateFutureRegionSupport",
            "Entity": {
                "Type": "AmiProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "FutureRegionSupport": {
                    "SupportedRegions": [
                        "All"
                    ]
```

```
                }
            }
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

渠道合作伙伴优惠

为任何产品类型创建 CPPO 草稿

以下代码示例演示如何为任何产品类型创建 CPPO 草稿，这样就可以在将它们发布给买家之前在内部进行审阅。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

创建采用合同定价的转售授权替代专属优惠

以下代码示例演示如何根据采用合同定价的现有协议创建转售授权替代专属优惠。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 RunChangesets。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateReplacementOfferUsingResaleAuthorization",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateReplacementOfferResaleAuth",
            "DetailsDocument": {
                "AgreementId": "agmt-11111111111111111111111",
                "ResaleAuthorizationId": "resaleauthz-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test replacement offer for SaaSProduct using AWS
 Marketplace API Reference Codes",
```

```
                "Description": "Test private resale replacement offer with contract
pricing for SaaSProduct"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Duration": "P12M",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 2
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementEndDate": "2024-01-30"
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "0"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOfferResaleAuth.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- 有关 API 的详细信息，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。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
```

```
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;

public class ListAllCppoOffers {

 /*
  * List all CPPOs created by a channel partner
  */
 public static void main(String[] args) {

  List<String> cppoOfferIds = getAllCppoOfferIds();

  ReferenceCodesUtils.formatOutput(cppoOfferIds);
 }

 public static List<String> getAllCppoOfferIds() {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  // get all offer entity ids
  List<String> entityIdList = new ArrayList<String>();

  ListEntitiesRequest listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_OFFER)
    .maxResults(10)
    .nextToken(null)
    .build();

  ListEntitiesResponse listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);

  for (EntitySummary entitySummary : listEntitiesResponse.entitySummaryList()) {
```

```
   entityIdList.add(entitySummary.entityId());
  }

  while (listEntitiesResponse.nextToken() != null) {
   listEntitiesRequest =
     ListEntitiesRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityType(ENTITY_TYPE_OFFER)
     .maxResults(10)
     .nextToken(listEntitiesResponse.nextToken())
     .build();
   listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);

   for (EntitySummary entitySummary : listEntitiesResponse.entitySummaryList()) {
    entityIdList.add(entitySummary.entityId());
   }
  }

  // filter for CPPO offers: ResaleAuthorizationId exists in Details

  List<String> cppoOfferIds = new ArrayList<String>();

  for (String entityId : entityIdList) {
   DescribeEntityRequest describeEntityRequest =
     DescribeEntityRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityId(entityId)
     .build();
   DescribeEntityResponse describeEntityResponse =
marketplaceCatalogClient.describeEntity(describeEntityRequest);

   Document resaleAuthorizationDocument =
describeEntityResponse.detailsDocument().asMap().get(ATTRIBUTE_RESALE_AUTHORIZATION_ID);
   String resaleAuthorizationId = resaleAuthorizationDocument != null ?
resaleAuthorizationDocument.asString() : "";

   if (!resaleAuthorizationId.isEmpty()) {
       cppoOfferIds.add(resaleAuthorizationId);
   }
  }
  return cppoOfferIds;
}
```

```
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListEntities中的。

列出渠道合作伙伴可获得的所有共享转售授权

以下代码示例演示如何列出渠道合作伙伴可获得的所有共享转售授权。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListEntities中的。

发布 CPPO 并附加买家 EULA

以下代码示例演示如何发布 CPPO 并附加买家 EULA。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 RunChangesets。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPOoffer",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111",
                "Name": "Test Offer",
                "Description":"Test product"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
           {
               "Type": "LegalTerm",
```

```
                "Documents": [
                 {
                  "Type": "CustomEula",
                  "Url": "https://s3.amazonaws.com/sample-bucket/custom-eula.pdf"
                 }
                ]
            }
        ]
          }
         },
         {
             "ChangeType": "UpdateTargeting",
             "Entity": {
                 "Type": "Offer@1.0",
                 "Identifier": "$CreateCPPOoffer.Entity.Identifier"
             },
             "DetailsDocument": {
                 "PositiveTargeting": {
                     "BuyerAccounts": ["222222222222"]
                     }
             }
         },
         {
             "ChangeType": "UpdateAvailability",
             "Entity": {
                 "Type": "Offer@1.0",
                 "Identifier": "$CreateCPPOoffer.Entity.Identifier"
             },
             "DetailsDocument": {
                 "AvailabilityEndDate": "2023-07-31"
                 }
         },
         {
             "ChangeType": "UpdateValidityTerms",
             "Entity": {
                 "Type": "Offer@1.0",
                 "Identifier": "$CreateCPPOoffer.Entity.Identifier"
             },
             "DetailsDocument": {
                 "Terms": [
                     {
                         "Type": "ValidityTerm",
                         "AgreementDuration": "P450D"
                     }
```

```
                ]
             }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

使用一次性转售授权发布 CPPO 并更新价格加成

以下代码示例演示如何对 AMI、SaaS 或容器产品使用一次性转售授权来发布 CPPO 并更新价格加成。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPOoffer",
            "DetailsDocument": {
```

```
            "ResaleAuthorizationId":"resaleauthz-1111111111111",
            "Name": "Test Offer",
            "Description":"Test product"
        }
    },
    {
        "ChangeType": "UpdateMarkup",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateCPPOoffer.Entity.Identifier"
        },
        "DetailsDocument": {
            "Percentage" : "5.0"
    }
    },
    {
        "ChangeType": "UpdateTargeting",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateCPPOoffer.Entity.Identifier"
        },
        "DetailsDocument": {
            "PositiveTargeting": {
                "BuyerAccounts": ["222222222222"]
                }
        }
    },
    {
        "ChangeType": "UpdateAvailability",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateCPPOoffer.Entity.Identifier"
        },
        "DetailsDocument": {
            "AvailabilityEndDate": "2023-07-31"
            }
    },
    {
        "ChangeType": "UpdateValidityTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateCPPOoffer.Entity.Identifier"
        },
        "DetailsDocument": {
```

```
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P450D"
                    }
                ]
             }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

发布 CPPO 草稿并更新价格加成

以下代码示例演示如何发布 CPPO 草稿和更新价格加成。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateOfferUsingResaleAuthorization",
            "Entity": {
```

```
            "Type": "Offer@1.0"
            },
            "ChangeName": "CreateCPPOoffer",
            "DetailsDocument": {
                "ResaleAuthorizationId":"resaleauthz-1111111111111",
                "Name": "Test Offer",
                "Description":"Test product"
            }
        },
        {
            "ChangeType": "UpdateMarkup",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Percentage" : "5.0"
        }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": ["222222222222"]
                    }
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-07-31"
                }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P450D"
                    }
                ]
             }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateCPPOoffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

更新 CPPO 的到期日期

以下代码示例演示如何更新 CPPO 的到期日期，让买家有更多时间评估和接受优惠。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 RunChangesets。

```
{
    "Catalog": "AWSMarketplace",
```

```
    "ChangeSet": [
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2025-07-31"
                }
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

容器产品

创建采用公开优惠草稿的容器产品草稿

以下代码示例演示如何创建采用公开优惠草稿的容器产品草稿。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

创建采用公开优惠和合同定价的受限容器产品

以下代码示例演示如何创建采用公开优惠、合同定价和标准 EULA 的受限容器产品。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 RunChangesets。

```
{
    "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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

实体

在单次调用中描述所有实体

以下代码示例演示如何在单次调用中描述所有实体。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
// 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考BatchDescribeEntities中的。

列出并描述与产品关联的所有优惠

以下代码示例演示如何列出并描述与产品关联的所有优惠。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
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《AWS SDK for Java 2.x API Reference》中的以下主题。
  - DescribeEntity
  - ListEntities

优惠

为 SaaS 产品创建自定义维度并创建专属优惠

以下代码示例演示如何为 SaaS 产品创建自定义维度并创建专属优惠。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
```

```
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": [
                {
                    "Types": [
                        "Entitled"
                    ],
                    "Description": "Custom Pricing 4 w/ terms and coverage to be
defined in Private Offer",
                    "Unit": "Units",
                    "Key": "Custom4",
                    "Name": "Custom Pricing 4"
                }
            ]
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            },
            "ChangeName": "CreateOfferChange"
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Private Test Offer - SaaS Contract Product",
                "Description": "Private Test Offer - SaaS Contract Product"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "Custom4",
                                        "Price": "300.0"
                                    }
                                ],
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P36M"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ],
    "ChangeSetName": "PrivateOfferWithCustomDimension"
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

为 AMI 或 SaaS 产品创建专属优惠草稿

以下代码示例演示如何为 AMI 或 SaaS 产品创建专属优惠草稿，以便您在向买家发布之前可以在内部对其进行审查。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "Test Private Offer"
            }
        }
    ]
}
```

- 有关 API 的详细信息，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 Pay-As-You-Go定价的私人报价

以下代码示例显示如何为 SaaS 产品创建包含合同和 Pay-As-You-Go定价的私募报价。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
 API Reference Code",
                "Description": "Test private offer with subscription pricing for
 SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
```

```
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
```

```
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

为 SaaS 产品创建采用合同定价和灵活付款时间表的专属优惠

以下代码示例演示如何为 SaaS 产品创建采用合同定价和灵活付款时间表的专属优惠。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
```

```
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with subscription pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
```

```
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 1
                            },
                            {
                                "DimensionKey": "PremiumService",
                                "MaxQuantity": 1
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P12M"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
```

```
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "200.00"
                            },
                            {
                                "ChargeDate": "2024-02-01",
                                "ChargeAmount": "170.00"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
```

```
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

为容器产品创建采用合同定价的专属优惠。

以下代码示例演示如何为容器产品创建采用合同定价的专属优惠。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
```

```
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for Container product using AWS
Marketplace API Reference Code",
                "Description": "Test private offer for Container product with
contract pricing using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
```

```
                                "Value": "P12M"
                            },
                            "Constraints": {
                                "MultipleDimensionSelection": "Disallowed",
                                "QuantityConfiguration": "Disallowed"
                            },
                            "RateCard": [
                                {
                                    "DimensionKey": "ReqPerHour",
                                    "Price": "0.25"
                                }
                            ]
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateLegalTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "LegalTerm",
                    "Documents": [
                        {
                            "Type": "StandardEula",
                            "Version": "2022-07-14"
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateAvailability",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
```

```
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

为 AMI 产品创建采用合同定价的专属优惠

以下代码示例演示如何为 AMI 产品创建采用合同定价的专属优惠。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 RunChangesets。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
```

```
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace API
Reference Code",
                "Description": "Test private offer with hourly annual pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
```

```
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "ReadOnlyUsers",
                                        "Price": "220.00"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
```

```
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

为 AMI 产品创建采用按小时按年定价和灵活付款时间表的专属优惠

以下代码示例演示如何为 AMI 产品创建采用按小时按年定价和灵活付款时间表的专属优惠。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
```

```
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace API
Reference Code",
                "Description": "Test private offer with hourly annual pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
```

```
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.17"
                                    }
                                ]
                            }
                        ]
```

```
                },
                {
                    "Type": "FixedUpfrontPricingTerm",
                    "CurrencyCode": "USD",
                    "Price": "0.0",
                    "Duration": "P365D",
                    "Grants": [
                        {
                            "DimensionKey": "t2.micro",
                            "MaxQuantity": 1
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateValidityTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "ValidityTerm",
                    "AgreementDuration": "P650D"
                }
            ]
        }
    },
    {
        "ChangeType": "UpdatePaymentScheduleTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "PaymentScheduleTerm",
                    "CurrencyCode": "USD",
                    "Schedule": [
                        {
```

```
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "200.00"
                            },
                            {
                                "ChargeDate": "2024-02-01",
                                "ChargeAmount": "170.00"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

为 AMI 产品创建采用按小时按年定价的专属优惠

以下代码示例演示如何为 AMI 产品创建采用按小时按年定价的专属优惠。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
```

```
   "ChangeSet": [
       {
           "ChangeType": "CreateOffer",
           "ChangeName": "CreateOfferChange",
           "Entity": {
               "Type": "Offer@1.0"
           },
           "DetailsDocument": {
               "ProductId": "prod-1111111111111"
           }
       },
       {
           "ChangeType": "UpdateInformation",
           "Entity": {
               "Type": "Offer@1.0",
               "Identifier": "$CreateOfferChange.Entity.Identifier"
           },
           "DetailsDocument": {
               "Name": "Test private offer for AmiProduct using AWS Marketplace API
Reference Code",
               "Description": "Test private offer with hourly annual pricing for
AmiProduct using AWS Marketplace API Reference Code"
           }
       },
       {
           "ChangeType": "UpdateTargeting",
           "Entity": {
               "Type": "Offer@1.0",
               "Identifier": "$CreateOfferChange.Entity.Identifier"
           },
           "DetailsDocument": {
               "PositiveTargeting": {
                   "BuyerAccounts": [
                       "111111111111",
                       "222222222222"
                   ]
               }
           }
       },
       {
           "ChangeType": "UpdateLegalTerms",
           "Entity": {
               "Type": "Offer@1.0",
               "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.micro",
                                        "Price": "0.17"
```

```
                                }
                            ]
                        }
                    ]
                },
                {
                    "Type": "ConfigurableUpfrontPricingTerm",
                    "CurrencyCode": "USD",
                    "RateCards": [
                        {
                            "Selector": {
                                "Type": "Duration",
                                "Value": "P365D"
                            },
                            "RateCard": [
                                {
                                    "DimensionKey": "t2.micro",
                                    "Price": "220.00"
                                }
                            ],
                            "Constraints": {
                                "MultipleDimensionSelection": "Allowed",
                                "QuantityConfiguration": "Allowed"
                            }
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateValidityTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "ValidityTerm",
                    "AgreementDuration": "P650D"
                }
            ]
        }
```

```
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

为 AMI 产品创建采用按小时定价的专属优惠

以下代码示例演示如何为 AMI 产品创建采用按小时定价的专属优惠。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
```

```
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for AmiProduct using AWS Marketplace API
Reference Code",
                "Description": "Test private offer with hourly pricing for
AmiProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
```

```
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateAvailability",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "AvailabilityEndDate": "2025-01-01"
        }
    },
    {
        "ChangeType": "UpdatePricingTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "PricingModel": "Usage",
            "Terms": [
                {
                    "Type": "UsageBasedPricingTerm",
                    "CurrencyCode": "USD",
                    "RateCards": [
                        {
                            "RateCard": [
                                {
                                    "DimensionKey": "t2.micro",
                                    "Price": "0.15"
                                }
                            ]
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateValidityTerms",
        "Entity": {
            "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P30D"
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

为 SaaS 产品创建采用订阅定价的专属优惠

以下代码示例演示如何为 SaaS 产品创建采用订阅定价的专属优惠。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
```

```
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
API Reference Code",
                "Description": "Test private offer with subscription pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.13"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.22"
                                    }
                                ]
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementDuration": "P30D"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[StartChangeSet](#)中的。

为 SaaS 产品创建采用分级合同定价的专属优惠

以下代码示例演示如何为 SaaS 产品创建采用分级合同定价的专属优惠。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test private offer for SaaSProduct using AWS Marketplace
 API Reference Code",
                "Description": "Test private offer with subscription pricing for
 SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
```

```
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "120.00"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "200.00"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Disallowed",
                                    "QuantityConfiguration": "Disallowed"
                                }
                            }
                        ]
```

```
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
```

```
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

为 SaaS 产品创建采用订阅定价的公开免费试用优惠

以下代码示例演示如何为 SaaS 产品创建采用订阅定价的公开免费试用优惠。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "prod-1111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public free trial offer for SaaSProduct using AWS
 Marketplace API Reference Code",
```

```
                "Description": "Test public free trial offer with subscription
pricing for SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Free",
                "Terms": [
                    {
                        "Type": "FreeTrialPricingTerm",
                        "Duration": "P20D",
                        "Grants": [
                            {
                                "DimensionKey": "WorkloadSmall"
                            },
                            {
                                "DimensionKey": "WorkloadMedium"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
```

```
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

创建采用合同定价的替代专属优惠

以下代码示例演示如何根据采用合同定价的现有协议创建替代专属优惠。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 RunChangesets。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType" : "CreateReplacementOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateReplacementOffer",
```

```
            "DetailsDocument": {
                "AgreementId": "agmt-11111111111111111111111"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test replacement offer for SaaSProduct using AWS
Marketplace API Reference Codes",
                "Description": "Test private replacement offer with contract pricing
for SaaSProduct"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "FixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.0",
                        "Grants": [
                            {
                                "DimensionKey": "BasicService",
                                "MaxQuantity": 2
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateValidityTerms",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "ValidityTerm",
                        "AgreementEndDate": "2024-01-30"
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePaymentScheduleTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "PaymentScheduleTerm",
                        "CurrencyCode": "USD",
                        "Schedule": [
                            {
                                "ChargeDate": "2024-01-01",
                                "ChargeAmount": "0"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
```

```
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-12-31"
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateReplacementOffer.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

描述公开优惠

以下代码示例演示如何描述公开优惠。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
// 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeEntity中的。

使专属优惠草稿过期

以下代码示例演示如何将专属优惠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优惠。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-01-01"
            }
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

列出所有专属优惠

以下代码示例演示如何列出所有专属优惠。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import software.amazon.awssdk.services.marketplacecatalog.model.EntityTypeFilters;
import software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferAvailabilityEndDateFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferAvailabilityEndDateFilterDate
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferBuyerAccountsFilter;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferFilters;
```

```
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferReleaseDateFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferReleaseDateFilterDateRange;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferTargetingFilter;

public class ListAllPrivateOffers {

 /*
  * List all my private offers and sort or filter them by Offer Publish Date, Offer
 Expiry Date and Buyer IDs
  *
  * OfferTargetingFilter = BuyerAccounts (private offer);
  * OfferBuyerAccountsFilter: Buyer IDs filter
  * OfferAvailabilityEndDateFilter : Offer Expiry Date filter
  * OfferReleaseDateFilter : Offer Publish Date filter
  */

 private static MarketplaceCatalogClient marketplaceCatalogClient =
   MarketplaceCatalogClient.builder()
   .httpClient(ApacheHttpClient.builder().build())
   .credentialsProvider(ProfileCredentialsProvider.create())
   .build();

 public static void main(String[] args) {

  String offerReleaseDateAfterValue = "2023-01-01T23:59:59Z";
  String offerAvailableEndDateAfterValue = "2040-12-24T23:59:59Z";

  List<EntitySummary> entitySummaryList =
 getEntitySummaryList(offerReleaseDateAfterValue, offerAvailableEndDateAfterValue);

  // for each offer id, output the offer detail using DescribeEntity API


  for (EntitySummary entitySummary : entitySummaryList) {
   DescribeEntityRequest describeEntityRequest =
     DescribeEntityRequest.builder()
     .catalog(AWS_MP_CATALOG)
     .entityId(entitySummary.entityId())
     .build();
   DescribeEntityResponse describeEntityResponse =
 marketplaceCatalogClient.describeEntity(describeEntityRequest);
```

```
  ReferenceCodesUtils.formatOutput(describeEntityResponse);
 }
}

public static List<EntitySummary> getEntitySummaryList (String
offerReleaseDateAfterValue, String offerAvailableEndDateAfterValue) {

 EntityTypeFilters entityTypeFilters =
   EntityTypeFilters.builder()
   .offerFilters(OfferFilters.builder()
     .targeting(OfferTargetingFilter.builder()
       .valueListWithStrings(OFFER_TARGETING_BUYERACCOUNTS)
       .build())
     .buyerAccounts(OfferBuyerAccountsFilter.builder()
       .wildCardValue(BUYER_ACCOUNT_ID)
       .build())
     .availabilityEndDate(OfferAvailabilityEndDateFilter.builder()
       .dateRange(OfferAvailabilityEndDateFilterDateRange.builder()
         .afterValue(offerAvailableEndDateAfterValue).build())
       .build())
     .releaseDate(OfferReleaseDateFilter.builder()
       .dateRange(OfferReleaseDateFilterDateRange.builder()
         .afterValue(offerReleaseDateAfterValue)
         .build())
       .build())
     .build())
   .build();

 ListEntitiesRequest listEntitiesRequest =
   ListEntitiesRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .entityType(ENTITY_TYPE_OFFER).maxResults(10)
   .entityTypeFilters(entityTypeFilters)
   .nextToken(null)
   .build();

 ListEntitiesResponse listEntitiesResponse =
marketplaceCatalogClient.listEntities(listEntitiesRequest);
 List<EntitySummary> entitySummaryList = new ArrayList<EntitySummary>();

 entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());

 while ( listEntitiesResponse.nextToken() != null &&
listEntitiesResponse.nextToken().length() > 0) {
```

```
    listEntitiesRequest =
      ListEntitiesRequest.builder()
      .catalog(AWS_MP_CATALOG)
      .entityType(ENTITY_TYPE_OFFER)
      .maxResults(10)
      .entityTypeFilters(entityTypeFilters)
      .nextToken(listEntitiesResponse.nextToken())
      .build();
    listEntitiesResponse =
  marketplaceCatalogClient.listEntities(listEntitiesRequest);
    entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());
   }

   return entitySummaryList;
  }

}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

列出针对特定产品 ID 发布的公开优惠和专属优惠

以下代码示例演示如何列出针对特定产品 ID 发布的公开优惠和专属优惠。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
package com.example.awsmarketplace.catalogapi;

import java.util.ArrayList;
import java.util.List;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
```

```
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import software.amazon.awssdk.services.marketplacecatalog.model.EntitySummary;
import software.amazon.awssdk.services.marketplacecatalog.model.EntityTypeFilters;
import software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.ListEntitiesResponse;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferFilters;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferProductIdFilter;
import software.amazon.awssdk.services.marketplacecatalog.model.OfferStateFilter;
import
 software.amazon.awssdk.services.marketplacecatalog.model.OfferTargetingFilter;

public class ListProductPublicOrPrivateReleasedOffers {

 /*
  * List released Public/Private offers for a specific product id.
  * Example below is to list released public offers.
  * To change to released private offers, change OFFER_TARGETING_NONE (None) to
 OFFER_TARGETING_BUYERACCOUNTS(BuyerAccounts)
  */
 public static void main(String[] args) {

  List<EntitySummary> entitySummaryList = getEntitySummaryLIst();
  ReferenceCodesUtils.formatOutput(entitySummaryList);
 }

 public static List<EntitySummary> getEntitySummaryLIst() {
  MarketplaceCatalogClient marketplaceCatalogClient =
    MarketplaceCatalogClient.builder()
    .httpClient(ApacheHttpClient.builder().build())
    .credentialsProvider(ProfileCredentialsProvider.create())
    .build();

  // define list entities filters

  EntityTypeFilters entityTypeFilters =
    EntityTypeFilters.builder()
    .offerFilters(OfferFilters.builder()
      .targeting(OfferTargetingFilter.builder()
        .valueListWithStrings(OFFER_TARGETING_NONE)
        .build())
      .state(OfferStateFilter.builder()
```

```
        .valueListWithStrings(OFFER_STATE_RELEASED)
        .build())
      .productId(OfferProductIdFilter.builder()
        .valueList(PRODUCT_ID)
        .build())
      .build())
    .build();

  ListEntitiesRequest listEntitiesRequest =
    ListEntitiesRequest.builder()
    .catalog(AWS_MP_CATALOG)
    .entityType(ENTITY_TYPE_OFFER)
    .maxResults(10)
    .entityTypeFilters(entityTypeFilters)
    .nextToken(null)
    .build();

  ListEntitiesResponse listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);

  // save all entitySummary of the results into entitySummaryList

  List<EntitySummary> entitySummaryList = new ArrayList<EntitySummary>();

  entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());

  while ( listEntitiesResponse.nextToken() != null &&
 listEntitiesResponse.nextToken().length() > 0) {
    listEntitiesRequest =
      ListEntitiesRequest.builder()
      .catalog(AWS_MP_CATALOG)
      .entityType(ENTITY_TYPE_OFFER)
      .maxResults(10)
      .entityTypeFilters(entityTypeFilters)
      .nextToken(listEntitiesResponse.nextToken())
      .build();
    listEntitiesResponse =
 marketplaceCatalogClient.listEntities(listEntitiesRequest);
    entitySummaryList.addAll(listEntitiesResponse.entitySummaryList());
  }
  return entitySummaryList;
 }

}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

更新报价以应用带 Pay-As-You-Go定价的合同

以下代码示例显示如何更新报价以应用带 Pay-As-You-Go定价的合同。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 RunChangesets。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
```

```
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

更新优惠以应用按小时按年定价

以下代码示例演示如何更新优惠以应用按小时按年定价。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "0.13"
                                    }
                                ]
                            }
                        ]
                    },
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
```

```
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P365D"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "m5.large",
                                        "Price": "20.03"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

更新优惠以将目标应用于特定地理区域

以下代码示例演示如何更新优惠以将目标应用于特定地理区域。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 RunChangesets。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "CountryCodes": [
                        "US",
                        "ES",
                        "FR",
                        "AU"
                    ]
                }
            }
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

更新公开优惠的名称和描述

以下代码示例演示如何更新公开优惠的名称和描述。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
```

```
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

更新优惠的 EULA

以下代码示例演示如何更新优惠的 EULA。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 RunChangesets。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Name": "New offer name",
                "Description": "New offer description"
            }
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

将专属优惠的到期日期更新为将来的日期

以下代码示例演示如何将专属优惠的到期日期更新为将来的日期，让买家有更多时间评估和接受优惠。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "Offer@1.0",
```

```
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2026-01-01"
            }
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

更新 SaaS 产品的公开免费试用优惠的免费试用期限

以下代码示例演示如何更新 SaaS 产品的公开免费试用优惠的免费试用期限。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "FreeTrialPricingTerm",
                        "Duration": "P21D",
                        "Grants": [
```

```
                            {
                                "DimensionKey": "WorkloadSmall"
                            },
                            {
                                "DimensionKey": "WorkloadMedium"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

更新优惠的退款政策

以下代码示例演示如何更新优惠的退款政策。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "offer-1111111111111"
            },
            "DetailsDocument": {
                "Terms": [
```

```
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Updated refund policy description"
                    }
                ]
            }
        }
    ]
}
```

- 有关 API 的详细信息，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

以下代码示例演示如何描述 AMI、SaaS 或容器产品，并检查它是否包含您想了解的有关该产品的所有信息。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
// 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeEntity中的。

列出所有 AMI、SaaS 或容器产品以及关联的公开优惠

以下代码示例演示如何列出所有 AMI、SaaS 或容器产品以及关联的公开优惠。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
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《AWS SDK for Java 2.x API Reference》中的以下主题。
  - DescribeEntity
  - ListEntities

转售授权

创建转售授权草稿

以下代码示例演示如何为任何产品类型创建转售授权草稿，这样您就可以在将其发布给渠道合作伙伴之前在内部进行审查。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 RunChangesets。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

描述转售授权

以下代码示例演示如何描述转售授权。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
// 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeEntity中的。

发布采用专属优惠的一次性转售授权

以下代码示例演示如何发布采用专属优惠的一次性转售授权，以便渠道合作伙伴可以使用它来创建渠道合作伙伴专属优惠（CPPO）。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
```

```
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
```

```
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

发布带到期日期的多次使用转售授权

以下代码示例演示如何为采用按小时按年定价的 AMI 产品发布带到期日期的多次使用转售授权，以便渠道合作伙伴可以使用它来创建 CPPO。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 RunChangesets。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
```

```
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
```

```
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

发布带到期日期的多次使用转售授权和 EULA

以下代码示例演示如何为任何产品类型发布带到期日期的多次使用转售授权，以及如何添加要发送给买家的自定义 EULA。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
```

```
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
```

```
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

发布带到期日期的多次使用转售授权和经销商合同文档

以下代码示例演示如何为任何产品类型发布带到期日期的多次使用转售授权，以及如何在 ISV 和渠道合作伙伴之间添加经销商合同文档。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    },
                    {
                        "Type": "ResaleLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomResellerContract",
                                "Url": "https://s3.amazonaws.com/aws-mp-standard-
contracts/Standard-Contact-for-AWS-Marketplace-2022-07-14.pdf"}
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
```

```
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

发布带到期日期的多次使用转售授权，并添加特定的买家账户

以下代码示例演示如何为任何产品类型发布带到期日期的多次使用转售授权，以及如何为转售添加特定的买家账户。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 RunChangesets。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "AvailabilityEndDate": "2023-05-31"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "111111111111"
                            ]
                        }
                    }
```

```
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

发布无到期日期的多次使用转售授权

以下代码示例演示如何为采用按小时按年定价的 AMI 产品发布无到期日期的多次使用转售授权，以便 CP 可以使用它来创建 CPPO。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 RunChangesets。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
```

```
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

发布无到期日期的多次使用转售授权和 EULA

以下代码示例演示如何为任何产品类型发布无到期日期的多次使用转售授权，以及如何添加要发送给买家的自定义 EULA。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
```

```
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
```

```
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

发布无到期日期的多次使用转售授权和经销商合同文档

以下代码示例演示如何为任何产品类型发布无到期日期的多次使用转售授权，以及如何在 ISV 和渠道合作伙伴之间添加经销商合同文档。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 RunChangesets。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
```

```
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
```

```
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    },
                    {
                        "Type": "ResaleLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomResellerContract",
                                "Url": "https://s3.amazonaws.com/aws-mp-standard-
contracts/Standard-Contact-for-AWS-Marketplace-2022-07-14.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

发布无到期日期的多次使用转售授权，并添加特定的买家账户

以下代码示例演示如何为任何产品类型发布无到期日期的多次使用转售授权，以及如何为转售添加特定的买家账户。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "111111111111"
                            ]
                        }
                    }
                ]
```

```
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- 有关 API 的详细信息，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

以下代码示例演示如何为任何产品类型发布一次性转售授权以及如何添加灵活的付款时间表。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 RunChangesets。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleFixedUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "Price": "0.00",
                        "Duration": "P12M",
                        "Grants": [
                          {
                            "DimensionKey": "Users",
                            "MaxQuantity": 10
                          }
```

```
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdatePaymentScheduleTerms",
        "Entity": {
            "Type": "ResaleAuthorization@1.0",
            "Identifier": "$ResaleAuthorization.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "ResalePaymentScheduleTerm",
                    "CurrencyCode": "USD",
                    "Schedule": [
                        {
                            "ChargeDate": "2023-09-01",
                            "ChargeAmount": "200.00"
                        },
                        {
                            "ChargeDate": "2023-12-01",
                            "ChargeAmount": "250.00"
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateAvailability",
        "Entity": {
            "Type": "ResaleAuthorization@1.0",
            "Identifier": "$ResaleAuthorization.Entity.Identifier"
        },
        "DetailsDocument": {
            "AvailabilityEndDate": "2023-06-30",
            "OffersMaxQuantity": 1
        }
    },
    {
        "ChangeType": "UpdateLegalTerms",
        "Entity": {
```

```
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- 有关 API 的详细信息，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为任何产品类型发布一次性转售授权，以及如何添加要发送给买家的自定义 EULA。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
```

```
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
```

```
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
```

```
}
```

- 有关 API 的详细信息，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

以下代码示例演示如何为任何产品类型发布一次性转售授权，以及如何为转售添加特定的买家账户。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 RunChangesets。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
```

```
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
```

```
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": "1"
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "111111111111"
                            ]
                        }
                    }
                ]
            }
        }
    ]
}
```

- 有关 API 的详细信息，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

以下代码示例演示如何为任何产品类型发布一次性转售授权，以及如何在 ISV 和渠道合作伙伴之间添加经销商合同文档。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 RunChangesets。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "ReleaseResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
```

```
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ResaleConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "t2.small",
                                        "Price": "150"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
```

```
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerLegalTerm",
                        "Documents": [
                            {
                                "Type": "CustomEula",
                                "Url": "https://s3.amazonaws.com/sample-bucket/
custom-eula.pdf"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}
```

- 有关 API 的详细信息，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它是否为续订

以下代码示例演示如何为任何产品类型发布一次性转售授权，以及添加它是否为续订。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 RunChangesets。

```
{
```

```
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateResaleAuthorization",
            "ChangeName": "ResaleAuthorization",
            "Entity": {
                "Type": "ResaleAuthorization@1.0"
            },
            "DetailsDocument": {
                "ProductId": "prod-1111111111111",
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "ResellerAccountId": "111111111111"
            }
        },
        {
            "ChangeType": "UpdateBuyerTargetingTerms",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "BuyerTargetingTerm",
                        "PositiveTargeting": {
                            "BuyerAccounts": [
                                "222222222222"
                            ]
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateAvailability",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "OffersMaxQuantity": 1
            }
        },
```

```
        {
            "ChangeType":"UpdateInformation",
            "Entity": {
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "$ResaleAuthorization.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product",
                "PreExistingBuyerAgreement": {
                    "AcquisitionChannel": "AwsMarketplace",
                    "PricingModel": "Contract"
                }
             }
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

限制转售授权

以下代码示例演示如何限制转售授权。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "RestrictResaleAuthorization",
            "Entity": {
```

```
                "Type": "ResaleAuthorization@1.0",
                "Identifier": "resaleauthz-1111111111111"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

更新一次性或多次使用转售授权的名称和描述

以下代码示例演示如何在为任何产品类型发布一次性或多次使用转售授权之前，更新此类授权的名称和描述。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType":"UpdateInformation",
            "Entity": {
             "Type": "ResaleAuthorization@1.0",
             "Identifier": "resaleauthz-1111111111111"
            },
            "DetailsDocument": {
             "Name": "TestResaleAuthorization",
                "Description": "Worldwide ResaleAuthorization for Test Product"
            }
        }
    ]
```

```
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

SaaS 产品

创建采用公开优惠草稿的 SaaS 产品草稿

以下代码示例演示如何创建采用公开优惠草稿的 SaaS 产品草稿。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

创建公开或受限 SaaS 产品和按合同定价的公开优惠

以下代码示例演示如何创建公开或受限 SaaS 产品和按合同定价的公开优惠。此示例创建标准或自定义 EULA。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
```

```
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
```

```
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                }
            ]
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
```

```
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P1M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
```

```
                                    "Price": "20"
                                },
                                {
                                    "DimensionKey": "PremiumService",
                                    "Price": "25"
                                }
                            ],
                            "Constraints": {
                                "MultipleDimensionSelection": "Allowed",
                                "QuantityConfiguration": "Allowed"
                            }
                        },
                        {
                            "Selector": {
                                "Type": "Duration",
                                "Value": "P12M"
                            },
                            "RateCard": [
                                {
                                    "DimensionKey": "BasicService",
                                    "Price": "150"
                                },
                                {
                                    "DimensionKey": "PremiumService",
                                    "Price": "300"
                                }
                            ],
                            "Constraints": {
                                "MultipleDimensionSelection": "Allowed",
                                "QuantityConfiguration": "Allowed"
                            }
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateLegalTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
```

```
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

创建具有定价合同的公开或有限的 SaaS 产品和公开报 Pay-As-You-Go价

以下代码示例演示如何使用定价合同创建公开或有限的 SaaS 产品和公开报 Pay-As-You-Go价。此示例创建标准或自定义 EULA。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
```

```
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
                            }
                        }
                    }
                ]
            }
```

```
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "WorkloadSmall",
                    "Description": "Workload: Per medium instance",
                    "Name": "Workload: Per medium instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "WorkloadMedium",
                    "Description": "Workload: Per large instance",
                    "Name": "Workload: Per large instance",
                    "Types": [
                        "ExternallyMetered"
                    ],
                    "Unit": "Units"
                }
```

```
            ]
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
```

```
                    "CurrencyCode": "USD",
                    "RateCards": [
                        {
                            "RateCard": [
                                {
                                    "DimensionKey": "WorkloadSmall",
                                    "Price": "0.15"
                                },
                                {
                                    "DimensionKey": "WorkloadMedium",
                                    "Price": "0.25"
                                }
                            ]
                        }
                    ]
                },
                {
                    "Type": "ConfigurableUpfrontPricingTerm",
                    "CurrencyCode": "USD",
                    "RateCards": [
                        {
                            "Selector": {
                                "Type": "Duration",
                                "Value": "P12M"
                            },
                            "RateCard": [
                                {
                                    "DimensionKey": "BasicService",
                                    "Price": "150"
                                },
                                {
                                    "DimensionKey": "PremiumService",
                                    "Price": "300"
                                }
                            ],
                            "Constraints": {
                                "MultipleDimensionSelection": "Allowed",
                                "QuantityConfiguration": "Allowed"
                            }
                        }
                    ]
                }
            ]
        }
```

```
        },
        {
            "ChangeType": "UpdateLegalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "LegalTerm",
                        "Documents": [
                            {
                                "Type": "StandardEula",
                                "Version": "2022-07-14"
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateSupportTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "SupportTerm",
                        "RefundPolicy": "Absolutely no refund, period."
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseOffer",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
```

```
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

创建公开或受限 SaaS 产品和采用订阅定价的公开优惠

以下代码示例演示如何创建公开或受限 SaaS 产品和采用订阅定价的公开优惠。此示例创建标准或自定义 EULA。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog": "AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "ChangeName": "CreateProductChange",
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
```

```
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://s3.amazonaws.com/logos/sample.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "UpdateTargeting",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PositiveTargeting": {
                    "BuyerAccounts": [
                        "111111111111",
                        "222222222222"
                    ]
                }
            }
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl":"https://sample.amazonaws.com/
sample-saas-fulfillment-url"
```

```
                        }
                    }
                }
            ]
        }
    },
    {
        "ChangeType": "AddDimensions",
        "Entity": {
            "Type": "SaaSProduct@1.0",
            "Identifier": "$CreateProductChange.Entity.Identifier"
        },
        "DetailsDocument": [
            {
                "Key": "WorkloadSmall",
                "Description": "Workload: Per medium instance",
                "Name": "Workload: Per medium instance",
                "Types": [
                    "ExternallyMetered"
                ],
                "Unit": "Units"
            },
            {
                "Key": "WorkloadMedium",
                "Description": "Workload: Per large instance",
                "Name": "Workload: Per large instance",
                "Types": [
                    "ExternallyMetered"
                ],
                "Unit": "Units"
            }
        ]
    },
    {
        "ChangeType": "ReleaseProduct",
        "Entity": {
            "Type": "SaaSProduct@1.0",
            "Identifier": "$CreateProductChange.Entity.Identifier"
        },
        "DetailsDocument": {}
    },
    {
        "ChangeType": "CreateOffer",
        "Entity": {
```

```
                "Type": "Offer@1.0"
            },
            "ChangeName": "CreateOfferChange",
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Name": "Test public offer for SaaSProduct using AWS Marketplace API
Reference Code",
                "Description": "Test public offer with contract pricing for
SaaSProduct using AWS Marketplace API Reference Code"
            }
        },
        {
            "ChangeType": "UpdatePricingTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Usage",
                "Terms": [
                    {
                        "Type": "UsageBasedPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "RateCard": [
                                    {
                                        "DimensionKey": "WorkloadSmall",
                                        "Price": "0.15"
                                    },
                                    {
                                        "DimensionKey": "WorkloadMedium",
                                        "Price": "0.25"
                                    }
                                ]
```

```
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateLegalTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "LegalTerm",
                    "Documents": [
                        {
                            "Type": "StandardEula",
                            "Version": "2022-07-14"
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateSupportTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "SupportTerm",
                    "RefundPolicy": "Absolutely no refund, period."
                }
            ]
        }
    },
    {
        "ChangeType": "ReleaseOffer",
        "Entity": {
```

```
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

发布 SaaS 产品和关联的公开优惠

以下代码示例演示如何发布 SaaS 产品和关联的公开优惠。默认情况下，产品将处于受限状态。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "CreateProduct",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "SaaSProduct@1.0"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
```

```
            "DetailsDocument": {
                "ProductTitle": "Sample product",
                "ShortDescription": "Brief description",
                "LongDescription": "Detailed description",
                "Highlights": [
                    "Sample highlight"
                ],
                "SearchKeywords": [
                    "Sample keyword"
                ],
                "Categories": [
                    "Data Catalogs"
                ],
                "LogoUrl": "https://bucketname.s3.amazonaws.com/logo.png",
                "VideoUrls": [
                    "https://sample.amazonaws.com/awsmp-video-1"
                ],
                "AdditionalResources": []
            }
        },
        {
            "ChangeType": "AddDimensions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": [
                {
                    "Key": "BasicService",
                    "Description": "Basic Service",
                    "Name": "Basic Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
                },
                {
                    "Key": "PremiumService",
                    "Description": "Premium Service",
                    "Name": "Premium Service",
                    "Types": [
                        "Entitled"
                    ],
                    "Unit": "Units"
```

```
                }
            ]
        },
        {
            "ChangeType": "AddDeliveryOptions",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "DeliveryOptions": [
                    {
                        "Details": {
                            "SaaSUrlDeliveryOptionDetails": {
                                "FulfillmentUrl": "https://www.aws.amazon.com/
marketplace/management"
                            }
                        }
                    }
                ]
            }
        },
        {
            "ChangeType": "ReleaseProduct",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "$CreateProductChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        },
        {
            "ChangeType": "CreateOffer",
            "ChangeName": "CreateOfferChange",
            "Entity": {
                "Type": "Offer@1.0"
            },
            "DetailsDocument": {
                "ProductId": "$CreateProductChange.Entity.Identifier"
            }
        },
        {
            "ChangeType": "UpdateInformation",
            "Entity": {
                "Type": "Offer@1.0",
```

```
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Name": "New Test Offer",
            "Description": "New offer description"
        }
    },
    {
        "ChangeType": "UpdateLegalTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "LegalTerm",
                    "Documents": [
                        {
                            "Type": "StandardEula",
                            "Version": "2022-07-14"
                        }
                    ]
                }
            ]
        }
    },
    {
        "ChangeType": "UpdateSupportTerms",
        "Entity": {
            "Type": "Offer@1.0",
            "Identifier": "$CreateOfferChange.Entity.Identifier"
        },
        "DetailsDocument": {
            "Terms": [
                {
                    "Type": "SupportTerm",
                    "RefundPolicy": "Updated refund policy description"
                }
            ]
        }
    },
    {
        "ChangeType": "UpdatePricingTerms",
```

```
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "PricingModel": "Contract",
                "Terms": [
                    {
                        "Type": "ConfigurableUpfrontPricingTerm",
                        "CurrencyCode": "USD",
                        "RateCards": [
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P1M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "20"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "25"
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            },
                            {
                                "Selector": {
                                    "Type": "Duration",
                                    "Value": "P12M"
                                },
                                "RateCard": [
                                    {
                                        "DimensionKey": "BasicService",
                                        "Price": "150"
                                    },
                                    {
                                        "DimensionKey": "PremiumService",
                                        "Price": "300"
```

```
                                    }
                                ],
                                "Constraints": {
                                    "MultipleDimensionSelection": "Allowed",
                                    "QuantityConfiguration": "Allowed"
                                }
                            }
                        ]
                    }
                ]
            }
        },
        {
            "ChangeType": "UpdateRenewalTerms",
            "Entity": {
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {
                "Terms": [
                    {
                        "Type": "RenewalTerm"
                    }
                ]
            }
        },
        {
            "ChangeType":"ReleaseOffer",
            "Entity":{
                "Type": "Offer@1.0",
                "Identifier": "$CreateOfferChange.Entity.Identifier"
            },
            "DetailsDocument": {}
        }
    ]
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

根据现有草稿发布 SaaS 产品和关联的公开优惠

以下代码示例演示如何根据现有草稿发布 SaaS 产品和关联的公开优惠。默认情况下，产品将处于受限状态。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
{
    "Catalog":"AWSMarketplace",
    "ChangeSet": [
        {
            "ChangeType": "UpdateVisibility",
            "ChangeName": "CreateProductChange",
            "Entity": {
                "Type": "SaaSProduct@1.0",
                "Identifier": "prod-1111111111111"
            },
            "DetailsDocument": {
                "TargetVisibility": "Public"
            }
        }
    ]
}
```

- 有关 API 的详细信息，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

以下代码示例演示如何更新 AMI 或 SaaS 产品的维度。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

要运行此示例，请将以下 JSON 更改集传递到实用程序部分的启动更改集的实用程序中的 `RunChangesets`。

```
{
    "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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

公用事业机构

用于启动更改集的实用程序

以下代码示例演示如何定义用于启动更改集的实用程序。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

用于从 JSON 文件加载更改集并开始处理更改集的实用程序。

```
package com.example.awsmarketplace.catalogapi;

import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

import org.apache.commons.io.IOUtils;
import org.apache.commons.lang3.StringUtils;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.document.Document;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.protocols.json.internal.unmarshall.document.DocumentUnmarshaller;
import software.amazon.awssdk.protocols.jsoncore.JsonNodeParser;
import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import software.amazon.awssdk.services.marketplacecatalog.model.Change;
import software.amazon.awssdk.services.marketplacecatalog.model.Entity;
import
 software.amazon.awssdk.services.marketplacecatalog.model.StartChangeSetRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.StartChangeSetResponse;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.ToNumberPolicy;
```

```
import com.example.awsmarketplace.catalogapi.Entity.ChangeSet;
import com.example.awsmarketplace.catalogapi.Entity.ChangeSetEntity;
import com.example.awsmarketplace.catalogapi.Entity.Root;
import com.example.awsmarketplace.utils.ReferenceCodesUtils;
import com.example.awsmarketplace.utils.StringSerializer;

/**
 * Before running this Java V2 code example, convert all Details attribute to
 DetailsDocument if any
 */

public class RunChangesets {

 private static final Gson GSON = new GsonBuilder()
   .setObjectToNumberStrategy(ToNumberPolicy.LAZILY_PARSED_NUMBER)
   .registerTypeAdapter(String.class, new StringSerializer())
   .create();

 public static void main(String[] args) {

  // input json can be specified here or passed from input parameter
  String inputChangeSetFile = "changeSets/offers/
CreateReplacementOfferFromAGWithContractPricingDetailDocument.json";

  if (args.length > 0)
   inputChangeSetFile = args[0];

  // parse the input changeset file to string for process
  String changeSetsInput = readChangeSetToString(inputChangeSetFile);

  // process the changeset request
  try {
   StartChangeSetResponse result = getChangeSetRequestResult(changeSetsInput);
   ReferenceCodesUtils.formatOutput(result);
  } catch (Exception e) {
   e.printStackTrace();
  }
 }

 public static StartChangeSetResponse getChangeSetRequestResult(String
 changeSetsInput) throws IOException {

  //set up AWS credentials
  MarketplaceCatalogClient marketplaceCatalogClient =
```

```
  MarketplaceCatalogClient.builder()
  .httpClient(ApacheHttpClient.builder().build())
  .credentialsProvider(ProfileCredentialsProvider.create())
  .build();

//changeset list to save all the changesets in the changesets file
List<Change> changeSetLists = new ArrayList<Change>();

// read all changesets into object
Root root = GSON.fromJson(changeSetsInput, Root.class);

// process each changeset and add each changeset request to changesets list
for (ChangeSet cs : root.changeSet) {

 ChangeSetEntity entity = cs.Entity;
 String entityType = entity.Type;
 String entityIdentifier = StringUtils.defaultIfBlank(entity.Identifier, null);
 Document detailsDocument = getDocumentFromObject(cs.DetailsDocument);

 Entity awsEntity =
   Entity.builder()
   .type(entityType)
   .identifier(entityIdentifier)
   .build();

 Change inputChangeRequest =
   Change.builder()
   .changeType(cs.ChangeType)
   .changeName(cs.ChangeName)
   .entity(awsEntity)
   .detailsDocument(detailsDocument)
   .build();

 changeSetLists.add(inputChangeRequest);
}

// process all changeset requests
StartChangeSetRequest startChangeSetRequest =
  StartChangeSetRequest.builder()
  .catalog(root.catalog)
  .changeSet(changeSetLists)
  .build();
```

```
 StartChangeSetResponse result =
marketplaceCatalogClient.startChangeSet(startChangeSetRequest);

 return result;
}

public static Document getDocumentFromObject(Object detailsObject) {

 String detailsString = "{}";
 try {
  detailsString = IOUtils.toString(new
ByteArrayInputStream(GSON.toJson(detailsObject).getBytes()), "UTF-8");
 } catch (IOException e) {
  e.printStackTrace();
 }

 JsonNodeParser jsonNodeParser = JsonNodeParser.create();
 Document doc = jsonNodeParser.parse(detailsString).visit(new
DocumentUnmarshaller());
 return doc;
}


public static String readChangeSetToString (String inputChangeSetFile) {

 InputStream changesetInputStream =
RunChangesets.class.getClassLoader().getResourceAsStream(inputChangeSetFile);

 String changeSetsInput = null;

 try {
  changeSetsInput = IOUtils.toString(changesetInputStream, "UTF-8");
 } catch (IOException e) {
  e.printStackTrace();
 }

 return changeSetsInput;

}
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartChangeSet中的。

AWS Marketplace 使用适用于 Java 的 SDK 2.x 的协议 API 示例

以下代码示例向您展示了如何使用 with Agreement API 执行操作和实现常见场景。 AWS SDK for Java 2.x AWS Marketplace

每个示例都包含一个指向完整源代码的链接，您可以从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。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
// 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SearchAgreements中的。

获取所有协议

以下代码示例演示如何获取所有协议。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
// 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SearchAgreements中的。

从协议中获取客户 ID

以下代码示例演示如何从协议中获取客户 ID。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
// 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[DescribeAgreement](#)中的。

从协议中获取财务详细信息

以下代码示例演示如何从协议中获取财务详细信息。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
// 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 的详细信息，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细信息

以下代码示例演示如何从协议中获取免费试用详细信息。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
// 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[DescribeAgreement](#)中的。

获取有关协议的信息

以下代码示例演示如何获取有关协议的信息。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
// 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 的详细信息，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细信息

以下代码示例演示如何从协议中获取产品和优惠详细信息。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
// 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeAgreement中的。

获取协议的 EULA

以下代码示例演示如何获取协议的 EULA。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
// 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[GetAgreementTerms](#)中的。

获取协议的自动续订条款

以下代码示例演示如何获取协议的自动续订条款。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
// 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetAgreementTerms中的。

获取在协议中购买的维度

以下代码示例演示如何获取在协议中购买的维度。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
// 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 的详细信息，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

以下代码示例演示如何获取在协议中购买的每个维度的实例。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
// 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 的详细信息，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

以下代码示例演示如何获取协议的付款时间表。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
// 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[GetAgreementTerms](#)中的。

获取协议中每个维度的定价

以下代码示例演示如何获取协议中每个维度的定价。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
// 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetAgreementTerms中的。

获取协议的定价类型

以下代码示例演示如何获取协议的定价类型。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
// SPDX-License-Identifier: Apache-2.0
package com.example.awsmarketplace.agreementapi;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import
 software.amazon.awssdk.services.marketplaceagreement.MarketplaceAgreementClient;
import software.amazon.awssdk.services.marketplaceagreement.model.AcceptedTerm;
import
 software.amazon.awssdk.services.marketplaceagreement.model.AgreementViewSummary;
import software.amazon.awssdk.services.marketplaceagreement.model.Filter;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.GetAgreementTermsResponse;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsRequest;
import
 software.amazon.awssdk.services.marketplaceagreement.model.SearchAgreementsResponse;

import com.fasterxml.jackson.annotation.JsonAutoDetect.Visibility;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.Objects;
import java.util.Set;

import org.apache.commons.lang3.tuple.Triple;

import software.amazon.awssdk.services.marketplacecatalog.MarketplaceCatalogClient;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityRequest;
import
 software.amazon.awssdk.services.marketplacecatalog.model.DescribeEntityResponse;

import static com.example.awsmarketplace.utils.ReferenceCodesConstants.*;
import com.fasterxml.jackson.annotation.PropertyAccessor;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.JsonNode;
```

```
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.ObjectWriter;
import com.fasterxml.jackson.datatype.jsr310.JavaTimeModule;

/*
 * Obtain the pricing type of the agreement (contract, FPS, metered, free etc.)
 */
public class GetAgreementPricingType {

 private static final String FILTER_NAME = "OfferId";

 private static final String FILTER_VALUE = OFFER_ID;

 // Product types
 private static final String SAAS_PRODUCT = "SaaSProduct";
 private static final String AMI_PRODUCT = "AmiProduct";
 private static final String ML_PRODUCT = "MachineLearningProduct";
 private static final String CONTAINER_PRODUCT = "ContainerProduct";
 private static final String DATA_PRODUCT = "DataProduct";
 private static final String PROSERVICE_PRODUCT = "ProfessionalServicesProduct";
 private static final String AIQ_PRODUCT = "AiqProduct";

 // Pricing types
 private static final String CCP = "CCP";
 private static final String ANNUAL = "Annual";
 private static final String CONTRACT = "Contract";
 private static final String SFT = "SaaS Free Trial";
 private static final String HMA = "Hourly and Monthly Agreements";
 private static final String HOURLY = "Hourly";
 private static final String MONTHLY = "Monthly";
 private static final String AFPS = "Annual FPS";
 private static final String CFPS = "Contract FPS";
 private static final String CCPFPS = "CCP with FPS";
 private static final String BYOL = "BYOL";
 private static final String FREE = "Free";
 private static final String FTH = "Free Trials and Hourly";

 // Agreement term pricing types
 private static final Set<String> LEGAL = Set.of("LegalTerm");
 private static final Set<String> CONFIGURABLE_UPFRONT =
 Set.of("ConfigurableUpfrontPricingTerm");
 private static final Set<String> USAGE_BASED = Set.of("UsageBasedPricingTerm");
 private static final Set<String> CONFIGURABLE_UPFRONT_AND_USAGE_BASED =
 Set.of("ConfigurableUpfrontPricingTerm", "UsageBasedPricingTerm");
```

```
private static final Set<String> FREE_TRIAL = Set.of("FreeTrialPricingTerm");
private static final Set<String> RECURRING_PAYMENT =
Set.of("RecurringPaymentTerm");
private static final Set<String> USAGE_BASED_AND_RECURRING_PAYMENT =
Set.of("UsageBasedPricingTerm", "RecurringPaymentTerm");
private static final Set<String> FIXED_UPFRONT_AND_PAYMENT_SCHEDULE =
Set.of("FixedUpfrontPricingTerm", "PaymentScheduleTerm");
private static final Set<String> FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED
= Set.of("FixedUpfrontPricingTerm", "PaymentScheduleTerm",
"UsageBasedPricingTerm");
private static final Set<String> BYOL_PRICING = Set.of("ByolPricingTerm");
private static final Set<String> FREE_TRIAL_AND_USAGE_BASED =
Set.of("FreeTrialPricingTerm", "UsageBasedPricingTerm");

private static final List<Set<String>> ALL_AGREEMENT_TERM_TYPES_COMBINATION
= Arrays.asList(LEGAL, CONFIGURABLE_UPFRONT, USAGE_BASED,
CONFIGURABLE_UPFRONT_AND_USAGE_BASED,
  FREE_TRIAL, RECURRING_PAYMENT, USAGE_BASED_AND_RECURRING_PAYMENT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, BYOL_PRICING,
FREE_TRIAL_AND_USAGE_BASED);

private static  MarketplaceAgreementClient marketplaceAgreementClient =
  MarketplaceAgreementClient.builder()
  .httpClient(ApacheHttpClient.builder().build())
  .credentialsProvider(ProfileCredentialsProvider.create())
  .build();

private static MarketplaceCatalogClient marketplaceCatalogClient =
  MarketplaceCatalogClient.builder()
  .httpClient(ApacheHttpClient.builder().build())
  .credentialsProvider(ProfileCredentialsProvider.create())
  .build();

   /*
    * Get agreement Pricing Type given product type, agreement term types and offer
types if needed
    */
public static String getPricingType(String productType, Set<String>
agreementTermType, Set<String> offerType) {
 Map<Triple<String, Set<String>, Set<String>>, String> pricingTypes = new
HashMap<>();
```

```
 pricingTypes.put(Triple.of(SAAS_PRODUCT, CONFIGURABLE_UPFRONT_AND_USAGE_BASED, new
HashSet<>()), CCP);
 pricingTypes.put(Triple.of(DATA_PRODUCT, CONFIGURABLE_UPFRONT_AND_USAGE_BASED, new
HashSet<>()), CCP);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT_AND_USAGE_BASED), ANNUAL);
 pricingTypes.put(Triple.of(AMI_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT_AND_USAGE_BASED), ANNUAL);
 pricingTypes.put(Triple.of(ML_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT_AND_USAGE_BASED), ANNUAL);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT), CONTRACT);
 pricingTypes.put(Triple.of(AMI_PRODUCT, CONFIGURABLE_UPFRONT,
CONFIGURABLE_UPFRONT), CONTRACT);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, CONFIGURABLE_UPFRONT, new HashSet<>()),
CONTRACT);
 pricingTypes.put(Triple.of(DATA_PRODUCT, CONFIGURABLE_UPFRONT, new HashSet<>()),
CONTRACT);
 pricingTypes.put(Triple.of(AIQ_PRODUCT, CONFIGURABLE_UPFRONT, new HashSet<>()),
CONTRACT);
 pricingTypes.put(Triple.of(PROSERVICE_PRODUCT, CONFIGURABLE_UPFRONT, new
HashSet<>()), CONTRACT);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, FREE_TRIAL, new HashSet<>()), SFT);
 pricingTypes.put(Triple.of(AMI_PRODUCT, USAGE_BASED_AND_RECURRING_PAYMENT, new
HashSet<>()), HMA);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, USAGE_BASED, new HashSet<>()), HOURLY);
 pricingTypes.put(Triple.of(AMI_PRODUCT, USAGE_BASED, new HashSet<>()), HOURLY);
 pricingTypes.put(Triple.of(ML_PRODUCT, USAGE_BASED, new HashSet<>()), HOURLY);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, RECURRING_PAYMENT, new HashSet<>()),
MONTHLY);
 pricingTypes.put(Triple.of(AMI_PRODUCT, RECURRING_PAYMENT, new HashSet<>()),
MONTHLY);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED), AFPS);
 pricingTypes.put(Triple.of(AMI_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED), AFPS);
 pricingTypes.put(Triple.of(ML_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE, new
HashSet<>()), AFPS);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
new HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(AMI_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE), CFPS);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE, new
HashSet<>()), CFPS);
```

```
 pricingTypes.put(Triple.of(DATA_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE, new
HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(AIQ_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE, new
HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(PROSERVICE_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE,
new HashSet<>()), CFPS);
 pricingTypes.put(Triple.of(SAAS_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, new HashSet<>()), CCPFPS);
 pricingTypes.put(Triple.of(DATA_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, new HashSet<>()), CCPFPS);
 pricingTypes.put(Triple.of(AIQ_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, new HashSet<>()), CCPFPS);
 pricingTypes.put(Triple.of(PROSERVICE_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE_AND_USAGE_BASED, new HashSet<>()), CCPFPS);
 pricingTypes.put(Triple.of(AMI_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(SAAS_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(PROSERVICE_PRODUCT, BYOL_PRICING, new HashSet<>()),
BYOL);
 pricingTypes.put(Triple.of(AIQ_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(ML_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, BYOL_PRICING, new HashSet<>()),
BYOL);
 pricingTypes.put(Triple.of(DATA_PRODUCT, BYOL_PRICING, new HashSet<>()), BYOL);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, LEGAL, new HashSet<>()), FREE);
 pricingTypes.put(Triple.of(AMI_PRODUCT, FREE_TRIAL_AND_USAGE_BASED, new
HashSet<>()), FTH);
 pricingTypes.put(Triple.of(CONTAINER_PRODUCT, FREE_TRIAL_AND_USAGE_BASED, new
HashSet<>()), FTH);
 pricingTypes.put(Triple.of(ML_PRODUCT, FREE_TRIAL_AND_USAGE_BASED, new
HashSet<>()), FTH);

 Triple<String, Set<String>, Set<String>> key = Triple.of(productType,
agreementTermType, offerType);

 if (pricingTypes.containsKey(key)) {
  return pricingTypes.get(key);
 } else {
  return "Unknown";
 }
}

/*
 * Given product type and agreement term types, some combinations need to check
offer term types as well.
```

```
 */
public static String needToCheckOfferTermsType(String productType, Set<String>
agreementTermTypes) {
 Map<KeyPair, String> offerTermTypes = new HashMap<>();
 offerTermTypes.put(new KeyPair(CONTAINER_PRODUCT, CONFIGURABLE_UPFRONT), "Y");
 offerTermTypes.put(new KeyPair(AMI_PRODUCT, CONFIGURABLE_UPFRONT), "Y");
 offerTermTypes.put(new KeyPair(CONTAINER_PRODUCT,
FIXED_UPFRONT_AND_PAYMENT_SCHEDULE), "Y");
 offerTermTypes.put(new KeyPair(AMI_PRODUCT, FIXED_UPFRONT_AND_PAYMENT_SCHEDULE),
"Y");

 KeyPair key = new KeyPair(productType, agreementTermTypes);
 if (offerTermTypes.containsKey(key)) {
  return offerTermTypes.get(key);
 } else {
  return null;
 }
}

public static List<AgreementViewSummary> getAgreementsById() {

 List<AgreementViewSummary> agreementSummaryList = new
ArrayList<AgreementViewSummary>();

 Filter partyType =
Filter.builder().name(PARTY_TYPE_FILTER_NAME).values(PARTY_TYPE_FILTER_VALUE_PROPOSER).buil

 Filter agreementType =
Filter.builder().name(AGREEMENT_TYPE_FILTER_NAME).values(AGREEMENT_TYPE_FILTER_VALUE_PURCHA

 Filter customizeFilter =
Filter.builder().name(FILTER_NAME).values(FILTER_VALUE).build();

 SearchAgreementsRequest searchAgreementsRequest =
   SearchAgreementsRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .filters(partyType, agreementType, customizeFilter).build();

 SearchAgreementsResponse searchResultResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);

 agreementSummaryList.addAll(searchResultResponse.agreementViewSummaries());
```

```
 while (searchResultResponse.nextToken() != null &&
searchResultResponse.nextToken().length() > 0) {
  searchAgreementsRequest =
SearchAgreementsRequest.builder().catalog(AWS_MP_CATALOG)
    .filters(partyType,
agreementType).nextToken(searchResultResponse.nextToken()).build();
  searchResultResponse =
marketplaceAgreementClient.searchAgreements(searchAgreementsRequest);
  agreementSummaryList.addAll(searchResultResponse.agreementViewSummaries());
 }
 return agreementSummaryList;

}

static class KeyPair {
 private final String first;
 private final Set<String> second;

 public KeyPair(String productType, Set<String> second) {
  this.first = productType;
  this.second = second;
 }

 @Override
 public int hashCode() {
  return Objects.hash(first, second);
 }

 @Override
 public boolean equals(Object obj) {
  if (this == obj)
   return true;
  if (obj == null || getClass() != obj.getClass())
   return false;
  KeyPair other = (KeyPair) obj;
  return Objects.equals(first, other.first) && Objects.equals(second,
other.second);
 }
}

/*
 * Get all the term types for the offer
 */
public static Set<String> getOfferTermTypes(String offerId) {
```

```
 Set<String> offerTermTypes = new HashSet<String>();

 DescribeEntityRequest request =
   DescribeEntityRequest.builder()
   .catalog(AWS_MP_CATALOG)
   .entityId(offerId)
   .build();

 DescribeEntityResponse result = marketplaceCatalogClient.describeEntity(request);

 String details = result.details();

 try {
  ObjectMapper objectMapper = new ObjectMapper();
  JsonNode rootNode = objectMapper.readTree(details);
  JsonNode termsNode = rootNode.get(ATTRIBUTE_TERMS);

  for (JsonNode termNode : termsNode) {
   if (termNode.get(ATTRIBUTE_TYPE_ENTITY) != null ) {
    offerTermTypes.add(termNode.get(ATTRIBUTE_TYPE_ENTITY).asText());
   }
  }
 } catch (Exception e) {
  e.printStackTrace();
 }

 return offerTermTypes;

}

/*
 * Get all the agreement term types
 */
public static Set<String> getAgreementTermTypes(GetAgreementTermsResponse
agreementTerm) {
 Set<String> agreementTermTypes = new HashSet<String>();
 try {
  for (AcceptedTerm term : agreementTerm.acceptedTerms()) {
   ObjectMapper objectMapper = new ObjectMapper();
   JsonNode termNode = objectMapper.readTree(getJson(term));
   Iterator<Map.Entry<String, JsonNode>> fieldsIterator = termNode.fields();
   while (fieldsIterator.hasNext()) {
    Map.Entry<String, JsonNode> entry = fieldsIterator.next();
```

```
    JsonNode value = entry.getValue();
    if (value.isObject() && value.has(ATTRIBUTE_TYPE_AGREEMENT)) {
     agreementTermTypes.add(value.get(ATTRIBUTE_TYPE_AGREEMENT).asText());
    }
   }
  }
 } catch (Exception e) {
  e.printStackTrace();
 }
 return agreementTermTypes;

}

/*
 * make sure all elements in array2 exist in array1
 */
public static boolean allElementsExist(Set<String> array1, Set<String> array2) {
 for (String element : array2) {
  boolean found = false;
  for (String str : array1) {
   if (element.equals(str)) {
    found = true;
    break;
   }
  }
  if (!found) {
   return false;
  }
 }
 return true;
}

/*
 * Find the combinations of the agreement term types for the agreement
 */
public static Set<String> getMatchedTermTypesCombination(Set<String>
agreementTermTypes) {
 Set<String> matchedCombination = new HashSet<String>();
 for (Set<String> element : ALL_AGREEMENT_TERM_TYPES_COMBINATION) {
  if (allElementsExist(agreementTermTypes, element)) {
   matchedCombination = element;
  }
 }
 return matchedCombination;
```

```
}

public static void main(String[] args) {

 List<AgreementViewSummary> agreements = getAgreementsById();

 for (AgreementViewSummary summary : agreements) {
  String pricingType = "";
  String agreementId = summary.agreementId();
  System.out.println(agreementId);
  String offerId = summary.proposalSummary().offerId();

  //get all pricing term types for the offer in the agreement
  Set<String> offerTermTypes = getOfferTermTypes(offerId);
  String productType = summary.proposalSummary().resources().get(0).type();

  //get all pricing term types for the agreement
  GetAgreementTermsRequest getAgreementTermsRequest =
    GetAgreementTermsRequest.builder().agreementId(agreementId)
    .build();
  GetAgreementTermsResponse getAgreementTermsResponse =
marketplaceAgreementClient.getAgreementTerms(getAgreementTermsRequest);
  Set<String> agreementTermTypes =
getAgreementTermTypes(getAgreementTermsResponse);

  //get matched pricing term type combination set
  Set<String> agreementMatchedTermType =
getMatchedTermTypesCombination(agreementTermTypes);

  //check to see if this agreement pricing term combination needs additional check
on offer pricing terms
  String needToCheckOfferType = needToCheckOfferTermsType(productType,
agreementMatchedTermType);

  // get the pricing type for the agreement based on the product type, agreement
term types and offer term types if needed
  if (needToCheckOfferType != null) {
   Set<String> offerMatchedTermType =
getMatchedTermTypesCombination(offerTermTypes);
   pricingType = getPricingType(productType, agreementMatchedTermType,
offerMatchedTermType);
  } else if (agreementMatchedTermType == LEGAL) {
   pricingType = FREE;
  } else {
```

```
    pricingType = getPricingType(productType, agreementMatchedTermType, new
 HashSet());
   }
   System.out.println("Pricing type is " + pricingType);
  }
 }

 private static String getJson(Object result) {
  String json = "";

  try {
   ObjectMapper om = new ObjectMapper();
   om.setVisibility(PropertyAccessor.FIELD, Visibility.ANY);
   om.registerModule(new JavaTimeModule());
   ObjectWriter ow = om.writer().withDefaultPrettyPrinter();

   json = ow.writeValueAsString(result);
  } catch (JsonProcessingException e) {
   e.printStackTrace();
  }
  return json;
 }

}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[DescribeAgreement](#)中的。

获取协议的产品类型

以下代码示例演示如何获取协议的产品类型。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
// 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeAgreement中的。

获取协议的状态

以下代码示例演示如何获取协议的状态。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
// 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeAgreement中的。

获取协议的支持条款

以下代码示例演示如何获取协议的支持条款。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
// 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetAgreementTerms中的。

获取协议的条款

以下代码示例演示如何获取协议的条款。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
// 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[GetAgreementTerms](#)中的。

按结束日期搜索协议

以下代码示例演示如何按结束日期搜索协议。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
// 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SearchAgreements中的。

通过一个自定义筛选条件来搜索协议

以下代码示例演示如何通过一个自定义筛选条件来搜索协议。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
// 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SearchAgreements中的。

通过两个自定义筛选条件来搜索协议

以下代码示例演示如何通过两个自定义筛选条件来搜索协议。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS Marketplace API Reference Code Library 存储库中查找完整示例，了解如何进行设置和运行。

```
// 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SearchAgreements中的。

MediaConvert 使用适用于 Java 的 SDK 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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从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`

以下代码示例演示了如何使用 `CreateJob`。

适用于 Java 的 SDK 2.x

> **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。

```
package com.example.mediaconvert;

import java.net.URI;
import java.util.HashMap;
import java.util.Map;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.Output;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroup;
```

```
import software.amazon.awssdk.services.mediaconvert.model.OutputGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.HlsGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.OutputGroupType;
import software.amazon.awssdk.services.mediaconvert.model.HlsDirectoryStructure;
import software.amazon.awssdk.services.mediaconvert.model.HlsManifestDurationFormat;
import software.amazon.awssdk.services.mediaconvert.model.HlsStreamInfResolution;
import software.amazon.awssdk.services.mediaconvert.model.HlsClientCache;
import software.amazon.awssdk.services.mediaconvert.model.HlsCaptionLanguageSetting;
import software.amazon.awssdk.services.mediaconvert.model.HlsManifestCompression;
import software.amazon.awssdk.services.mediaconvert.model.HlsCodecSpecification;
import software.amazon.awssdk.services.mediaconvert.model.HlsOutputSelection;
import software.amazon.awssdk.services.mediaconvert.model.HlsProgramDateTime;
import software.amazon.awssdk.services.mediaconvert.model.HlsTimedMetadataId3Frame;
import software.amazon.awssdk.services.mediaconvert.model.HlsSegmentControl;
import software.amazon.awssdk.services.mediaconvert.model.FileGroupSettings;
import software.amazon.awssdk.services.mediaconvert.model.ContainerSettings;
import software.amazon.awssdk.services.mediaconvert.model.VideoDescription;
import software.amazon.awssdk.services.mediaconvert.model.ContainerType;
import software.amazon.awssdk.services.mediaconvert.model.ScalingBehavior;
import software.amazon.awssdk.services.mediaconvert.model.VideoTimecodeInsertion;
import software.amazon.awssdk.services.mediaconvert.model.ColorMetadata;
import software.amazon.awssdk.services.mediaconvert.model.RespondToAfd;
import software.amazon.awssdk.services.mediaconvert.model.AfdSignaling;
import software.amazon.awssdk.services.mediaconvert.model.DropFrameTimecode;
import software.amazon.awssdk.services.mediaconvert.model.VideoCodecSettings;
import software.amazon.awssdk.services.mediaconvert.model.H264Settings;
import software.amazon.awssdk.services.mediaconvert.model.VideoCodec;
import software.amazon.awssdk.services.mediaconvert.model.CreateJobRequest;
import software.amazon.awssdk.services.mediaconvert.model.H264RateControlMode;
import software.amazon.awssdk.services.mediaconvert.model.H264QualityTuningLevel;
import software.amazon.awssdk.services.mediaconvert.model.H264SceneChangeDetect;
import
 software.amazon.awssdk.services.mediaconvert.model.AacAudioDescriptionBroadcasterMix;
import software.amazon.awssdk.services.mediaconvert.model.H264ParControl;
import software.amazon.awssdk.services.mediaconvert.model.AacRawFormat;
import software.amazon.awssdk.services.mediaconvert.model.H264QvbrSettings;
import
 software.amazon.awssdk.services.mediaconvert.model.H264FramerateConversionAlgorithm;
import software.amazon.awssdk.services.mediaconvert.model.H264CodecLevel;
import software.amazon.awssdk.services.mediaconvert.model.H264FramerateControl;
import software.amazon.awssdk.services.mediaconvert.model.AacCodingMode;
import software.amazon.awssdk.services.mediaconvert.model.H264Telecine;
import
 software.amazon.awssdk.services.mediaconvert.model.H264FlickerAdaptiveQuantization;
```

```
import software.amazon.awssdk.services.mediaconvert.model.H264GopSizeUnits;
import software.amazon.awssdk.services.mediaconvert.model.H264CodecProfile;
import software.amazon.awssdk.services.mediaconvert.model.H264GopBReference;
import software.amazon.awssdk.services.mediaconvert.model.AudioTypeControl;
import software.amazon.awssdk.services.mediaconvert.model.AntiAlias;
import software.amazon.awssdk.services.mediaconvert.model.H264SlowPal;
import
 software.amazon.awssdk.services.mediaconvert.model.H264SpatialAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.H264Syntax;
import software.amazon.awssdk.services.mediaconvert.model.M3u8Settings;
import software.amazon.awssdk.services.mediaconvert.model.InputDenoiseFilter;
import
 software.amazon.awssdk.services.mediaconvert.model.H264TemporalAdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.CreateJobResponse;
import
 software.amazon.awssdk.services.mediaconvert.model.H264UnregisteredSeiTimecode;
import software.amazon.awssdk.services.mediaconvert.model.H264EntropyEncoding;
import software.amazon.awssdk.services.mediaconvert.model.InputPsiControl;
import software.amazon.awssdk.services.mediaconvert.model.ColorSpace;
import software.amazon.awssdk.services.mediaconvert.model.H264RepeatPps;
import software.amazon.awssdk.services.mediaconvert.model.H264FieldEncoding;
import software.amazon.awssdk.services.mediaconvert.model.M3u8NielsenId3;
import software.amazon.awssdk.services.mediaconvert.model.InputDeblockFilter;
import software.amazon.awssdk.services.mediaconvert.model.InputRotate;
import software.amazon.awssdk.services.mediaconvert.model.H264DynamicSubGop;
import software.amazon.awssdk.services.mediaconvert.model.TimedMetadata;
import software.amazon.awssdk.services.mediaconvert.model.JobSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioDefaultSelection;
import software.amazon.awssdk.services.mediaconvert.model.VideoSelector;
import software.amazon.awssdk.services.mediaconvert.model.AacSpecification;
import software.amazon.awssdk.services.mediaconvert.model.Input;
import software.amazon.awssdk.services.mediaconvert.model.OutputSettings;
import software.amazon.awssdk.services.mediaconvert.model.H264AdaptiveQuantization;
import software.amazon.awssdk.services.mediaconvert.model.AudioLanguageCodeControl;
import software.amazon.awssdk.services.mediaconvert.model.InputFilterEnable;
import software.amazon.awssdk.services.mediaconvert.model.AudioDescription;
import software.amazon.awssdk.services.mediaconvert.model.H264InterlaceMode;
import software.amazon.awssdk.services.mediaconvert.model.AudioCodecSettings;
import software.amazon.awssdk.services.mediaconvert.model.AacSettings;
import software.amazon.awssdk.services.mediaconvert.model.AudioCodec;
import software.amazon.awssdk.services.mediaconvert.model.AacRateControlMode;
import software.amazon.awssdk.services.mediaconvert.model.AacCodecProfile;
import software.amazon.awssdk.services.mediaconvert.model.HlsIFrameOnlyManifest;
import software.amazon.awssdk.services.mediaconvert.model.FrameCaptureSettings;
```

```
import software.amazon.awssdk.services.mediaconvert.model.AudioSelector;
import software.amazon.awssdk.services.mediaconvert.model.M3u8PcrControl;
import software.amazon.awssdk.services.mediaconvert.model.InputTimecodeSource;
import software.amazon.awssdk.services.mediaconvert.model.HlsSettings;
import software.amazon.awssdk.services.mediaconvert.model.M3u8Scte35Source;

/**
 * Create a MediaConvert job. Must supply MediaConvert access role Amazon
 * Resource Name (ARN), and a
 * valid video input file via Amazon S3 URL.
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 */
public class CreateJob {
        public static void main(String[] args) {
                final String usage = """

                                Usage:
                                    <mcRoleARN> <fileInput>\s

                                Where:
                                    mcRoleARN - The MediaConvert Role ARN.\s
                                    fileInput -  The URL of an Amazon S3 bucket
 where the input file is located.\s
                                """;

                if (args.length != 2) {
                        System.out.println(usage);
                        System.exit(1);
                }

                String mcRoleARN = args[0];
                String fileInput = args[1];
                Region region = Region.US_WEST_2;
                MediaConvertClient mc = MediaConvertClient.builder()
                                .region(region)
                                .build();

                String id = createMediaJob(mc, mcRoleARN, fileInput);
```

```
                System.out.println("MediaConvert job created. Job Id = " + id);
                mc.close();
        }

        public static String createMediaJob(MediaConvertClient mc, String mcRoleARN,
 String fileInput) {

                String s3path = fileInput.substring(0, fileInput.lastIndexOf('/') +
 1) + "javasdk/out/";
                String fileOutput = s3path + "index";
                String thumbsOutput = s3path + "thumbs/";
                String mp4Output = s3path + "mp4/";

                try {
                        System.out.println("MediaConvert role arn: " + mcRoleARN);
                        System.out.println("MediaConvert input file: " + fileInput);
                        System.out.println("MediaConvert output path: " + s3path);

                        // output group Preset HLS low profile
                        Output hlsLow = createOutput("hls_low", "_low", "_$dt$",
 750000, 7, 1920, 1080, 640);
                        // output group Preset HLS media profile
                        Output hlsMedium = createOutput("hls_medium", "_medium", "_
$dt$", 1200000, 7, 1920, 1080, 1280);
                        // output group Preset HLS high profole
                        Output hlsHigh = createOutput("hls_high", "_high", "_$dt$",
 3500000, 8, 1920, 1080, 1920);

                        OutputGroup appleHLS = OutputGroup.builder().name("Apple
 HLS").customName("Example")

 .outputGroupSettings(OutputGroupSettings.builder()

 .type(OutputGroupType.HLS_GROUP_SETTINGS)

 .hlsGroupSettings(HlsGroupSettings.builder()

 .directoryStructure(

    HlsDirectoryStructure.SINGLE_DIRECTORY)

 .manifestDurationFormat(

    HlsManifestDurationFormat.INTEGER)
```

```
.streamInfResolution(
   HlsStreamInfResolution.INCLUDE)
.clientCache(HlsClientCache.ENABLED)
.captionLanguageSetting(
   HlsCaptionLanguageSetting.OMIT)
.manifestCompression(
   HlsManifestCompression.NONE)
.codecSpecification(
   HlsCodecSpecification.RFC_4281)
.outputSelection(
   HlsOutputSelection.MANIFESTS_AND_SEGMENTS)
.programDateTime(HlsProgramDateTime.EXCLUDE)
.programDateTimePeriod(600)
.timedMetadataId3Frame(
   HlsTimedMetadataId3Frame.PRIV)
.timedMetadataId3Period(10)
.destination(fileOutput)
.segmentControl(HlsSegmentControl.SEGMENTED_FILES)
.minFinalSegmentLength((double) 0)
.segmentLength(4).minSegmentLength(0).build())
                                                    .build())
                                        .outputs(hlsLow, hlsMedium,
hlsHigh).build();
```

```
                        OutputGroup fileMp4 = OutputGroup.builder().name("File
Group").customName("mp4")

.outputGroupSettings(OutputGroupSettings.builder()

.type(OutputGroupType.FILE_GROUP_SETTINGS)

.fileGroupSettings(FileGroupSettings.builder()

.destination(mp4Output).build())
                                                          .build())
                                          .outputs(Output.builder().extension("mp4")

.containerSettings(ContainerSettings.builder()

.container(ContainerType.MP4).build())

.videoDescription(VideoDescription.builder().width(1280)
                                                                        .height(720)

.scalingBehavior(ScalingBehavior.DEFAULT)

.sharpness(50).antiAlias(AntiAlias.ENABLED)

.timecodeInsertion(

   VideoTimecodeInsertion.DISABLED)

.colorMetadata(ColorMetadata.INSERT)

.respondToAfd(RespondToAfd.NONE)

.afdSignaling(AfdSignaling.NONE)

.dropFrameTimecode(DropFrameTimecode.ENABLED)

.codecSettings(VideoCodecSettings.builder()

   .codec(VideoCodec.H_264)

   .h264Settings(H264Settings

                    .builder()
```

```
                .rateControlMode(
                        H264RateControlMode.QVBR)
                .parControl(H264ParControl.INITIALIZE_FROM_SOURCE)
                .qualityTuningLevel(
                        H264QualityTuningLevel.SINGLE_PASS)
                .qvbrSettings(
                        H264QvbrSettings.builder()
                                        .qvbrQualityLevel(
                                                        8)
                                        .build())
                .codecLevel(H264CodecLevel.AUTO)
                .codecProfile(H264CodecProfile.MAIN)
                .maxBitrate(2400000)
                .framerateControl(
                        H264FramerateControl.INITIALIZE_FROM_SOURCE)
                .gopSize(2.0)
                .gopSizeUnits(H264GopSizeUnits.SECONDS)
                .numberBFramesBetweenReferenceFrames(
                        2)
                .gopClosedCadence(
                        1)
                .gopBReference(H264GopBReference.DISABLED)
```

```
                .slowPal(H264SlowPal.DISABLED)
                .syntax(H264Syntax.DEFAULT)
                .numberReferenceFrames(
                        3)
                .dynamicSubGop(H264DynamicSubGop.STATIC)
                .fieldEncoding(H264FieldEncoding.PAFF)
                .sceneChangeDetect(
                        H264SceneChangeDetect.ENABLED)
                .minIInterval(0)
                .telecine(H264Telecine.NONE)
                .framerateConversionAlgorithm(
                        H264FramerateConversionAlgorithm.DUPLICATE_DROP)
                .entropyEncoding(
                        H264EntropyEncoding.CABAC)
                .slices(1)
                .unregisteredSeiTimecode(
                        H264UnregisteredSeiTimecode.DISABLED)
                .repeatPps(H264RepeatPps.DISABLED)
                .adaptiveQuantization(
                        H264AdaptiveQuantization.HIGH)
                .spatialAdaptiveQuantization(
                        H264SpatialAdaptiveQuantization.ENABLED)
```

```
                    .temporalAdaptiveQuantization(
                                    H264TemporalAdaptiveQuantization.ENABLED)
                    .flickerAdaptiveQuantization(
                                    H264FlickerAdaptiveQuantization.DISABLED)
                    .softness(0)
                    .interlaceMode(H264InterlaceMode.PROGRESSIVE)
                    .build())
    .build())
                                                                    .build())
.audioDescriptions(AudioDescription.builder()
.audioTypeControl(AudioTypeControl.FOLLOW_INPUT)
.languageCodeControl(
   AudioLanguageCodeControl.FOLLOW_INPUT)
.codecSettings(AudioCodecSettings.builder()
    .codec(AudioCodec.AAC)
    .aacSettings(AacSettings
                    .builder()
                    .codecProfile(AacCodecProfile.LC)
                    .rateControlMode(
                                    AacRateControlMode.CBR)
                    .codingMode(AacCodingMode.CODING_MODE_2_0)
                    .sampleRate(44100)
```

```
                    .bitrate(160000)

                    .rawFormat(AacRawFormat.NONE)

                    .specification(AacSpecification.MPEG4)

                    .audioDescriptionBroadcasterMix(

                                    AacAudioDescriptionBroadcasterMix.NORMAL)

                    .build())

   .build())
                                                                    .build())
                                                        .build())
                                        .build();
                        OutputGroup thumbs = OutputGroup.builder().name("File
Group").customName("thumbs")

.outputGroupSettings(OutputGroupSettings.builder()

.type(OutputGroupType.FILE_GROUP_SETTINGS)

.fileGroupSettings(FileGroupSettings.builder()

.destination(thumbsOutput).build())
                                                        .build())
                                        .outputs(Output.builder().extension("jpg")

.containerSettings(ContainerSettings.builder()

.container(ContainerType.RAW).build())

.videoDescription(VideoDescription.builder()

.scalingBehavior(ScalingBehavior.DEFAULT)

.sharpness(50).antiAlias(AntiAlias.ENABLED)

.timecodeInsertion(

   VideoTimecodeInsertion.DISABLED)
```

```
.colorMetadata(ColorMetadata.INSERT)

.dropFrameTimecode(DropFrameTimecode.ENABLED)

.codecSettings(VideoCodecSettings.builder()

   .codec(VideoCodec.FRAME_CAPTURE)

   .frameCaptureSettings(

                  FrameCaptureSettings

                                  .builder()

                                  .framerateNumerator(

                                                  1)

                                  .framerateDenominator(

                                                  1)

                                  .maxCaptures(10000000)

                                  .quality(80)

                                  .build())

   .build())
                                                                        .build())
                                                  .build())
                                      .build();

                      Map<String, AudioSelector> audioSelectors = new HashMap<>();
                      audioSelectors.put("Audio Selector 1",

AudioSelector.builder().defaultSelection(AudioDefaultSelection.DEFAULT)
                                                          .offset(0).build());

                      JobSettings jobSettings =
JobSettings.builder().inputs(Input.builder()
                                      .audioSelectors(audioSelectors)
                                      .videoSelector(
```

```
VideoSelector.builder().colorSpace(ColorSpace.FOLLOW)

.rotate(InputRotate.DEGREE_0).build())

.filterEnable(InputFilterEnable.AUTO).filterStrength(0)
                                        .deblockFilter(InputDeblockFilter.DISABLED)

.denoiseFilter(InputDenoiseFilter.DISABLED).psiControl(InputPsiControl.USE_PSI)

.timecodeSource(InputTimecodeSource.EMBEDDED).fileInput(fileInput).build())
                                        .outputGroups(appleHLS, thumbs,
fileMp4).build();

                        CreateJobRequest createJobRequest =
CreateJobRequest.builder().role(mcRoleARN)
                                        .settings(jobSettings)
                                        .build();

                        CreateJobResponse createJobResponse =
mc.createJob(createJobRequest);
                        return createJobResponse.job().id();

                } catch (MediaConvertException e) {
                        System.out.println(e.toString());
                        System.exit(0);
                }
                return "";
        }

        private final static Output createOutput(String customName,
                        String nameModifier,
                        String segmentModifier,
                        int qvbrMaxBitrate,
                        int qvbrQualityLevel,
                        int originWidth,
                        int originHeight,
                        int targetWidth) {

                int targetHeight = Math.round(originHeight * targetWidth /
originWidth)
                                - (Math.round(originHeight * targetWidth /
originWidth) % 4);
                Output output = null;
```

```
                try {
                        output =
Output.builder().nameModifier(nameModifier).outputSettings(OutputSettings.builder()

.hlsSettings(HlsSettings.builder().segmentModifier(segmentModifier)

.audioGroupId("program_audio")

.iFrameOnlyManifest(HlsIFrameOnlyManifest.EXCLUDE).build())
                                        .build())

.containerSettings(ContainerSettings.builder().container(ContainerType.M3_U8)

.m3u8Settings(M3u8Settings.builder().audioFramesPerPes(4)

.pcrControl(M3u8PcrControl.PCR_EVERY_PES_PACKET)

.pmtPid(480).privateMetadataPid(503)

.programNumber(1).patInterval(0).pmtInterval(0)

.scte35Source(M3u8Scte35Source.NONE)

.scte35Pid(500).nielsenId3(M3u8NielsenId3.NONE)

.timedMetadata(TimedMetadata.NONE)

.timedMetadataPid(502).videoPid(481)

.audioPids(482, 483, 484, 485, 486, 487, 488,

   489, 490, 491, 492)
                                                                        .build())
                                                    .build())
                                        .videoDescription(

VideoDescription.builder().width(targetWidth)

.height(targetHeight)

.scalingBehavior(ScalingBehavior.DEFAULT)

.sharpness(50).antiAlias(AntiAlias.ENABLED)
```

```
.timecodeInsertion(
   VideoTimecodeInsertion.DISABLED)
.colorMetadata(ColorMetadata.INSERT)
.respondToAfd(RespondToAfd.NONE)
.afdSignaling(AfdSignaling.NONE)
.dropFrameTimecode(DropFrameTimecode.ENABLED)
.codecSettings(VideoCodecSettings.builder()
   .codec(VideoCodec.H_264)
   .h264Settings(H264Settings
                  .builder()
                  .rateControlMode(
                                 H264RateControlMode.QVBR)
                  .parControl(H264ParControl.INITIALIZE_FROM_SOURCE)
                  .qualityTuningLevel(
                                 H264QualityTuningLevel.SINGLE_PASS)
                  .qvbrSettings(H264QvbrSettings
                                 .builder()
                                 .qvbrQualityLevel(
                                                qvbrQualityLevel)
                                 .build())
                  .codecLevel(H264CodecLevel.AUTO)
                  .codecProfile((targetHeight > 720
```

```
                        && targetWidth > 1280)
                                        ? H264CodecProfile.HIGH
                                        : H264CodecProfile.MAIN)
                .maxBitrate(qvbrMaxBitrate)
                .framerateControl(
                        H264FramerateControl.INITIALIZE_FROM_SOURCE)
                .gopSize(2.0)
                .gopSizeUnits(H264GopSizeUnits.SECONDS)
                .numberBFramesBetweenReferenceFrames(
                        2)
                .gopClosedCadence(
                        1)
                .gopBReference(H264GopBReference.DISABLED)
                .slowPal(H264SlowPal.DISABLED)
                .syntax(H264Syntax.DEFAULT)
                .numberReferenceFrames(
                        3)
                .dynamicSubGop(H264DynamicSubGop.STATIC)
                .fieldEncoding(H264FieldEncoding.PAFF)
                .sceneChangeDetect(
                        H264SceneChangeDetect.ENABLED)
                .minIInterval(0)
```

```
                    .telecine(H264Telecine.NONE)
                    .framerateConversionAlgorithm(
                                    H264FramerateConversionAlgorithm.DUPLICATE_DROP)
                    .entropyEncoding(
                                    H264EntropyEncoding.CABAC)
                    .slices(1)
                    .unregisteredSeiTimecode(
                                    H264UnregisteredSeiTimecode.DISABLED)
                    .repeatPps(H264RepeatPps.DISABLED)
                    .adaptiveQuantization(
                                    H264AdaptiveQuantization.HIGH)
                    .spatialAdaptiveQuantization(
                                    H264SpatialAdaptiveQuantization.ENABLED)
                    .temporalAdaptiveQuantization(
                                    H264TemporalAdaptiveQuantization.ENABLED)
                    .flickerAdaptiveQuantization(
                                    H264FlickerAdaptiveQuantization.DISABLED)
                    .softness(0)
                    .interlaceMode(H264InterlaceMode.PROGRESSIVE)
                    .build())
    .build())
                                                                    .build())
```

```
 .audioDescriptions(AudioDescription.builder()

 .audioTypeControl(AudioTypeControl.FOLLOW_INPUT)

 .languageCodeControl(AudioLanguageCodeControl.FOLLOW_INPUT)

 .codecSettings(AudioCodecSettings.builder()

 .codec(AudioCodec.AAC).aacSettings(AacSettings

    .builder()

    .codecProfile(AacCodecProfile.LC)

    .rateControlMode(

                    AacRateControlMode.CBR)

    .codingMode(AacCodingMode.CODING_MODE_2_0)

    .sampleRate(44100)

    .bitrate(96000)

    .rawFormat(AacRawFormat.NONE)

    .specification(AacSpecification.MPEG4)

    .audioDescriptionBroadcasterMix(

                    AacAudioDescriptionBroadcasterMix.NORMAL)

    .build())
                                                                        .build())
                                                          .build())
                                       .build();
               } catch (MediaConvertException e) {
                       e.printStackTrace();
                       System.exit(0);
               }
               return output;
        }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateJob中的。

`GetJob`

以下代码示例演示了如何使用 `GetJob`。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.GetJobRequest;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.GetJobResponse;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import java.net.URI;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetJob {

    public static void main(String[] args) {

        final String usage = "\n" +
                "  <jobId> \n\n" +
                "Where:\n" +
                "  jobId - The job id value.\n\n";
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String jobId = args[0];
        Region region = Region.US_WEST_2;
        MediaConvertClient mc = MediaConvertClient.builder()
                .region(region)
                .build();

        getSpecificJob(mc, jobId);
        mc.close();
    }

    public static void getSpecificJob(MediaConvertClient mc, String jobId) {
        try {
            GetJobRequest jobRequest = GetJobRequest.builder()
                    .id(jobId)
                    .build();

            GetJobResponse response = mc.getJob(jobRequest);
            System.out.println("The ARN of the job is " + response.job().arn());

        } catch (MediaConvertException e) {
            System.out.println(e.toString());
            System.exit(0);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetJob中的。

ListJobs

以下代码示例演示了如何使用 ListJobs。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.mediaconvert.MediaConvertClient;
import software.amazon.awssdk.services.mediaconvert.model.ListJobsRequest;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsResponse;
import software.amazon.awssdk.services.mediaconvert.model.DescribeEndpointsRequest;
import software.amazon.awssdk.services.mediaconvert.model.ListJobsResponse;
import software.amazon.awssdk.services.mediaconvert.model.Job;
import software.amazon.awssdk.services.mediaconvert.model.MediaConvertException;
import java.net.URI;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListJobs {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MediaConvertClient mc = MediaConvertClient.builder()
                .region(region)
                .build();

        listCompleteJobs(mc);
        mc.close();
    }

    public static void listCompleteJobs(MediaConvertClient mc) {
        try {
            // Create the ListJobsRequest
            ListJobsRequest jobsRequest = ListJobsRequest.builder()
```

```
                    .maxResults(10)
                    .status("COMPLETE")
                    .build();

            // Call the listJobs operation
            ListJobsResponse jobsResponse = mc.listJobs(jobsRequest);
            List<Job> jobs = jobsResponse.jobs();
            for (Job job : jobs) {
                System.out.println("The JOB ARN is : " + job.arn());
            }

        } catch (MediaConvertException e) {
            System.out.println(e.toString());
            System.exit(0);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListJobs中的。

使用 SDK for 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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从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`

以下代码示例演示了如何使用 `DeleteProgressUpdateStream`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[DeleteProgressUpdateStream](#)中的。

`DescribeApplicationState`

以下代码示例演示了如何使用 `DescribeApplicationState`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeApplicationState中的。

`DescribeMigrationTask`

以下代码示例演示了如何使用 `DescribeMigrationTask`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeMigrationTask中的。

`ImportMigrationTask`

以下代码示例演示了如何使用 `ImportMigrationTask`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ImportMigrationTask中的。

`ListApplications`

以下代码示例演示了如何使用 `ListApplications`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ListApplications](#)中的。

ListCreatedArtifacts

以下代码示例演示了如何使用 ListCreatedArtifacts。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.migrationhub.MigrationHubClient;
import software.amazon.awssdk.services.migrationhub.model.CreatedArtifact;
import
 software.amazon.awssdk.services.migrationhub.model.ListCreatedArtifactsRequest;
import
 software.amazon.awssdk.services.migrationhub.model.ListCreatedArtifactsResponse;
import software.amazon.awssdk.services.migrationhub.model.MigrationHubException;
import java.util.List;

/**
 * To run this Java V2 code example, ensure that you have setup your development
 * environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListCreatedArtifacts {
    public static void main(String[] args) {
        Region region = Region.US_WEST_2;
        MigrationHubClient migrationClient = MigrationHubClient.builder()
                .region(region)
                .build();

        listArtifacts(migrationClient);
        migrationClient.close();
    }

    public static void listArtifacts(MigrationHubClient migrationClient) {
        try {
```

```
            ListCreatedArtifactsRequest listCreatedArtifactsRequest =
 ListCreatedArtifactsRequest.builder()
                    .maxResults(10)
                    .migrationTaskName("SampleApp5")
                    .progressUpdateStream("ProgressSteamB")
                    .build();

            ListCreatedArtifactsResponse response =
 migrationClient.listCreatedArtifacts(listCreatedArtifactsRequest);
            List<CreatedArtifact> apps = response.createdArtifactList();
            for (CreatedArtifact artifact : apps) {
                System.out.println("APp Id is " + artifact.description());
                System.out.println("The name is " + artifact.name());
            }

        } catch (MigrationHubException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListCreatedArtifacts中的。

`ListMigrationTasks`

以下代码示例演示了如何使用 `ListMigrationTasks`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListMigrationTasks中的。

使用适用于 Java 的 SDK 2.x 的 Amazon MSK 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 MSK 配合使用来执行操作和实现常见场景。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 无服务器示例

无服务器示例

通过 Amazon MSK 触发器调用 Lambda 函数

以下代码示例演示如何实现一个 Lambda 函数，该函数接收通过接收来自 Amazon MSK 集群的记录而触发的事件。该函数检索 MSK 有效负载，并记录下记录内容。

适用于 Java 的 SDK 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。

通过 Java 将 Amazon MSK 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.KafkaEvent;
import com.amazonaws.services.lambda.runtime.events.KafkaEvent.KafkaEventRecord;
```

```
import java.util.Base64;
import java.util.Map;

public class Example implements RequestHandler<KafkaEvent, Void> {

    @Override
    public Void handleRequest(KafkaEvent event, Context context) {
        for (Map.Entry<String, java.util.List<KafkaEventRecord>> entry :
 event.getRecords().entrySet()) {
            String key = entry.getKey();
            System.out.println("Key: " + key);

            for (KafkaEventRecord record : entry.getValue()) {
                System.out.println("Record: " + record);

                byte[] value = Base64.getDecoder().decode(record.getValue());
                String message = new String(value);
                System.out.println("Message: " + message);
            }
        }

        return null;
    }
}
```

使用适用于 Java 的 SDK 2.x 的 Neptune 示例

以下代码示例向您展示了如何使用与 Neptune AWS SDK for Java 2.x 配合使用来执行操作和实现常见场景。

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 基本功能
- 操作
- 场景

开始使用

Hello Neptune

以下代码示例显示了如何开始使用 Neptune。

适用于 Java 的 SDK 2.x

> **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。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloNeptune {
    public static void main(String[] args) {
        NeptuneAsyncClient neptuneClient = NeptuneAsyncClient.create();
        describeDbCluster(neptuneClient).join(); // This ensures the async code runs
 to completion
    }

    /**
     * Describes the Amazon Neptune DB clusters.
     *
     * @param neptuneClient the Neptune asynchronous client used to make the request
```

```
    * @return a {@link CompletableFuture} that completes when the operation is
finished
    */
   public static CompletableFuture<Void> describeDbCluster(NeptuneAsyncClient
neptuneClient) {
       DescribeDbClustersRequest request = DescribeDbClustersRequest.builder()
               .maxRecords(20)
               .build();

       SdkPublisher<DescribeDbClustersResponse> paginator =
neptuneClient.describeDBClustersPaginator(request);
       CompletableFuture<Void> future = new CompletableFuture<>();

       paginator.subscribe(new Subscriber<DescribeDbClustersResponse>() {
           private Subscription subscription;

           @Override
           public void onSubscribe(Subscription s) {
               this.subscription = s;
               s.request(Long.MAX_VALUE); // request all items
           }

           @Override
           public void onNext(DescribeDbClustersResponse response) {
               response.dbClusters().forEach(cluster -> {
                   System.out.println("Cluster Identifier: " +
cluster.dbClusterIdentifier());
                   System.out.println("Status: " + cluster.status());
               });
           }

           @Override
           public void onError(Throwable t) {
               future.completeExceptionally(t);
           }

           @Override
           public void onComplete() {
               future.complete(null);
           }
       });

       return future.whenComplete((result, throwable) -> {
           neptuneClient.close();
```

```
            if (throwable != null) {
                System.err.println("Error describing DB clusters: " +
throwable.getMessage());
            }
        });
    }
```

- 有关 API 的详细信息，请参阅《AWS SDK for Java 2.x AP I 参考》中的 “描述DBClusters分页器”。

基本功能

了解基本功能

以下代码示例展示了如何：

- 创建 Amazon Neptune 子网组。
- 创建 Neptune 集群。
- 创建 Neptune 实例。
- 检查 Neptune 实例的状态。
- 显示 Neptune 集群详细信息。
- 停止 Neptune 集群。
- 启动 Neptune 集群。
- 删除 Neptune 资产。

适用于 Java 的 SDK 2.x

> **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。

运行一个交互式场景，演示 Neptune 的功能。

```
public class NeptuneScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
```

```
   private static final Logger logger =
LoggerFactory.getLogger(NeptuneScenario.class);
   static Scanner scanner = new Scanner(System.in);
   static NeptuneActions neptuneActions = new NeptuneActions();

   public static void main(String[] args) {
       final String usage =
               """
               Usage:
                   <subnetGroupName> <clusterName> <dbInstanceId>

               Where:
                   subnetGroupName - The name of an existing Neptune DB subnet
group that includes subnets in at least two Availability Zones.
                   clusterName     - The unique identifier for the Neptune DB
cluster.
                   dbInstanceId    - The identifier for a specific Neptune DB
instance within the cluster.
               """;
       String subnetGroupName = "neptuneSubnetGroup65";
       String clusterName = "neptuneCluster65";
       String dbInstanceId = "neptuneDB65";

       logger.info("""
                  Amazon Neptune is a fully managed graph
                  database service by AWS, designed specifically
                  for handling complex relationships and connected
                  datasets at scale. It supports two popular graph models:
                  property graphs (via openCypher and Gremlin) and RDF
                  graphs (via SPARQL). This makes Neptune ideal for
                  use cases such as knowledge graphs, fraud detection,
                  social networking, recommendation engines, and
                  network management, where relationships between
                  entities are central to the data.

                  Being fully managed, Neptune handles database
                  provisioning, patching, backups, and replication,
                  while also offering high availability and durability
                  within AWS's infrastructure.

                  For developers, programming with Neptune allows
                  for building intelligent, relationship-aware
                  applications that go beyond traditional tabular
                  databases. Developers can use the AWS SDK for Java
```

```
                    to automate infrastructure operations (via NeptuneClient).

                     Let's get started...
                     """);
        waitForInputToContinue(scanner);
        runScenario(subnetGroupName, dbInstanceId, clusterName);
    }

    public static void runScenario(String subnetGroupName, String dbInstanceId,
String clusterName) {
        logger.info(DASHES);
        logger.info("1. Create a Neptune DB Subnet Group");
        logger.info("The Neptune DB subnet group is used when launching a Neptune
cluster");
        waitForInputToContinue(scanner);
        try {
            neptuneActions.createSubnetGroupAsync(subnetGroupName).join();

        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ServiceQuotaExceededException) {
                logger.error("The request failed due to service quota exceeded: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Create a Neptune Cluster");
        logger.info("A Neptune Cluster allows you to store and query highly
connected datasets with low latency.");
        waitForInputToContinue(scanner);
        String dbClusterId;
        try {
            dbClusterId = neptuneActions.createDBClusterAsync(clusterName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ServiceQuotaExceededException) {
                logger.error("The request failed due to service quota exceeded: {}",
cause.getMessage());
```

```
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }

        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("3. Create a Neptune DB Instance");
        logger.info("In this step, we add a new database instance to the Neptune
cluster");
        waitForInputToContinue(scanner);
        try {
        neptuneActions.createDBInstanceAsync(dbInstanceId, dbClusterId).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ServiceQuotaExceededException) {
                logger.error("The request failed due to service quota exceeded: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Check the status of the Neptune DB Instance");
        logger.info("""
                    In this step, we will wait until the DB instance
                    becomes available. This may take around 10 minutes.
                    """);
        waitForInputToContinue(scanner);
        try {
            neptuneActions.checkInstanceStatus(dbInstanceId, "available").join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            logger.error("An unexpected error occurred.", cause);
            return;
        }
        waitForInputToContinue(scanner);
```

```
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5.Show Neptune Cluster details");
        waitForInputToContinue(scanner);
        try {
            neptuneActions.describeDBClustersAsync(clusterName).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The request failed due to the resource not found: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. Stop the Amazon Neptune cluster");
        logger.info("""
                    Once stopped, this step polls the status
                    until the cluster is in a stopped state.
                    """);
        waitForInputToContinue(scanner);
        try {
            neptuneActions.stopDBClusterAsync(dbClusterId);
            neptuneActions.waitForClusterStatus(dbClusterId, "stopped");
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The request failed due to the resource not found: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
```

```
        logger.info("7. Start the Amazon Neptune cluster");
        logger.info("""
                    Once started, this step polls the clusters
                    status until it's in an available state.
                    We will also poll the instance status.
                    """);
        waitForInputToContinue(scanner);
        try {
            neptuneActions.startDBClusterAsync(dbClusterId);
            neptuneActions.waitForClusterStatus(dbClusterId, "available");
            neptuneActions.checkInstanceStatus(dbInstanceId, "available").join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof ResourceNotFoundException) {
                logger.error("The request failed due to the resource not found: {}",
cause.getMessage());
            } else {
                logger.error("An unexpected error occurred.", cause);
            }
            return;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Delete the Neptune Assets");
        logger.info("Would you like to delete the Neptune Assets? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete the Neptune assets.");
            try {
                neptuneActions.deleteNeptuneResourcesAsync(dbInstanceId,
clusterName, subnetGroupName);
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof ResourceNotFoundException) {
                    logger.error("The request failed due to the resource not found:
{}", cause.getMessage());
                } else {
                    logger.error("An unexpected error occurred.", cause);
                }
                return;
            }
        } else {
            logger.info("You selected not to delete Neptune assets.");
```

```
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info(
                """
                Thank you for checking out the Amazon Neptune Service Use demo. We
 hope you
                learned something new, or got some inspiration for your own apps
 today.
                For more AWS code examples, have a look at:
                https://docs.aws.amazon.com/code-library/latest/ug/what-is-code-
library.html
                """);
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Neptune SDK 方法的包装器类。

```
public class NeptuneActions {
    private CompletableFuture<Void> instanceCheckFuture;
    private static NeptuneAsyncClient neptuneAsyncClient;
    private final Region region = Region.US_EAST_1;
```

```
   private static final Logger logger =
LoggerFactory.getLogger(NeptuneActions.class);
   private final NeptuneClient neptuneClient =
NeptuneClient.builder().region(region).build();

   /**
    * Retrieves an instance of the NeptuneAsyncClient.
    * <p>
    * This method initializes and returns a singleton instance of the
NeptuneAsyncClient. The client
    * is configured with the following settings:
    * <ul>
    *     <li>Maximum concurrency: 100</li>
    *     <li>Connection timeout: 60 seconds</li>
    *     <li>Read timeout: 60 seconds</li>
    *     <li>Write timeout: 60 seconds</li>
    *     <li>API call timeout: 2 minutes</li>
    *     <li>API call attempt timeout: 90 seconds</li>
    *     <li>Retry strategy: STANDARD</li>
    * </ul>
    * The client is built using the NettyNioAsyncHttpClient.
    *
    * @return the singleton instance of the NeptuneAsyncClient
    */
   private static NeptuneAsyncClient getAsyncClient() {
       if (neptuneAsyncClient == null) {
           SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                   .maxConcurrency(100)
                   .connectionTimeout(Duration.ofSeconds(60))
                   .readTimeout(Duration.ofSeconds(60))
                   .writeTimeout(Duration.ofSeconds(60))
                   .build();

           ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                   .apiCallTimeout(Duration.ofMinutes(2))
                   .apiCallAttemptTimeout(Duration.ofSeconds(90))
                   .retryStrategy(RetryMode.STANDARD)
                   .build();

           neptuneAsyncClient = NeptuneAsyncClient.builder()
                   .httpClient(httpClient)
                   .overrideConfiguration(overrideConfig)
                   .build();
```

```
        }
        return neptuneAsyncClient;
    }

    /**
     * Asynchronously deletes a set of Amazon Neptune resources in a defined order.
     * <p>
     * The method performs the following operations in sequence:
     * <ol>
     *     <li>Deletes the Neptune DB instance identified by {@code dbInstanceId}.</
li>
     *     <li>Waits until the DB instance is fully deleted.</li>
     *     <li>Deletes the Neptune DB cluster identified by {@code dbClusterId}.</
li>
     *     <li>Deletes the Neptune DB subnet group identified by {@code
 subnetGroupName}.</li>
     * </ol>
     * <p>
     * If any step fails, the subsequent operations are not performed, and the
 exception
     * is logged. This method blocks the calling thread until all operations
 complete.
     *
     * @param dbInstanceId      the ID of the Neptune DB instance to delete
     * @param dbClusterId       the ID of the Neptune DB cluster to delete
     * @param subnetGroupName   the name of the Neptune DB subnet group to delete
     */
    public void deleteNeptuneResourcesAsync(String dbInstanceId, String dbClusterId,
 String subnetGroupName) {
        deleteDBInstanceAsync(dbInstanceId)
                .thenCompose(v -> waitUntilInstanceDeletedAsync(dbInstanceId))
                .thenCompose(v -> deleteDBClusterAsync(dbClusterId))
                .thenCompose(v -> deleteDBSubnetGroupAsync(subnetGroupName))
                .whenComplete((v, ex) -> {
                    if (ex != null) {
                        logger.info("Failed to delete Neptune resources: " +
 ex.getMessage());
                    } else {
                        logger.info("Neptune resources deleted successfully.");
                    }
                })
                .join(); // Waits for the entire async chain to complete
    }
```

```
    /**
     * Deletes a subnet group.
     *
     * @param subnetGroupName the identifier of the subnet group to delete
     * @return a {@link CompletableFuture} that completes when the cluster has been
deleted
     */
    public CompletableFuture<Void> deleteDBSubnetGroupAsync(String subnetGroupName)
{
        DeleteDbSubnetGroupRequest request = DeleteDbSubnetGroupRequest.builder()
                .dbSubnetGroupName(subnetGroupName)
                .build();

        return getAsyncClient().deleteDBSubnetGroup(request)
                .thenAccept(response -> logger.info("## Deleting Subnet Group: " +
subnetGroupName));
    }

    /**
     * Deletes a DB instance asynchronously.
     *
     * @param clusterId the identifier of the cluster to delete
     * @return a {@link CompletableFuture} that completes when the cluster has been
deleted
     */
    public CompletableFuture<Void> deleteDBClusterAsync(String clusterId) {
        DeleteDbClusterRequest request = DeleteDbClusterRequest.builder()
                .dbClusterIdentifier(clusterId)
                .skipFinalSnapshot(true)
                .build();

        return getAsyncClient().deleteDBCluster(request)
                .thenAccept(response -> System.out.println("## Deleting DB Cluster:
" + clusterId));
    }

    public CompletableFuture<Void> waitUntilInstanceDeletedAsync(String instanceId)
{
        CompletableFuture<Void> future = new CompletableFuture<>();
        long startTime = System.currentTimeMillis();
        checkInstanceDeletedRecursive(instanceId, startTime, future);
        return future;
    }
```

```
   /**
    * Deletes a DB instance asynchronously.
    *
    * @param instanceId the identifier of the DB instance to be deleted
    * @return a {@link CompletableFuture} that completes when the DB instance has
been deleted
    */
   public CompletableFuture<Void> deleteDBInstanceAsync(String instanceId) {
       DeleteDbInstanceRequest request = DeleteDbInstanceRequest.builder()
               .dbInstanceIdentifier(instanceId)
               .skipFinalSnapshot(true)
               .build();

       return getAsyncClient().deleteDBInstance(request)
               .thenAccept(response -> System.out.println("## Deleting DB Instance:
" + instanceId));
   }


   private void checkInstanceDeletedRecursive(String instanceId, long startTime,
CompletableFuture<Void> future) {
       DescribeDbInstancesRequest request = DescribeDbInstancesRequest.builder()
               .dbInstanceIdentifier(instanceId)
               .build();

       getAsyncClient().describeDBInstances(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof NeptuneException &&
                               ((NeptuneException)
cause).awsErrorDetails().errorCode().equals("DBInstanceNotFound")) {
                           long elapsed = (System.currentTimeMillis() -
startTime) / 1000;
                           logger.info("\r Instance %s deleted after %ds%n",
instanceId, elapsed);
                           future.complete(null);
                           return;
                       }
                       future.completeExceptionally(new CompletionException("Error
polling DB instance", cause));
                       return;
                   }
```

```
                    String status =
response.dbInstances().get(0).dbInstanceStatus();
                    long elapsed = (System.currentTimeMillis() - startTime) / 1000;
                    System.out.printf("\r  Waiting: Instance %s status: %-10s (%ds
elapsed)", instanceId, status, elapsed);
                    System.out.flush();

                    CompletableFuture.delayedExecutor(20, TimeUnit.SECONDS)
                            .execute(() -> checkInstanceDeletedRecursive(instanceId,
startTime, future));
                });
    }


    public void waitForClusterStatus(String clusterId, String desiredStatus) {
        System.out.printf("Waiting for cluster '%s' to reach status '%s'...\n",
clusterId, desiredStatus);
        CompletableFuture<Void> future = new CompletableFuture<>();
        checkClusterStatusRecursive(clusterId, desiredStatus,
System.currentTimeMillis(), future);
        future.join();
    }

    private void checkClusterStatusRecursive(String clusterId, String desiredStatus,
long startTime, CompletableFuture<Void> future) {
        DescribeDbClustersRequest request = DescribeDbClustersRequest.builder()
                .dbClusterIdentifier(clusterId)
                .build();

        getAsyncClient().describeDBClusters(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        future.completeExceptionally(
                                new CompletionException("Error checking Neptune
cluster status", cause)
                        );
                        return;
                    }

                    List<DBCluster> clusters = response.dbClusters();
                    if (clusters.isEmpty()) {
                        future.completeExceptionally(new RuntimeException("Cluster
not found: " + clusterId));
```

```
                        return;
                    }

                    String currentStatus = clusters.get(0).status();
                    long elapsedSeconds = (System.currentTimeMillis() - startTime) /
1000;
                    System.out.printf("\r Elapsed: %-20s  Cluster status: %-20s",
formatElapsedTime((int) elapsedSeconds), currentStatus);
                    System.out.flush();

                    if (desiredStatus.equalsIgnoreCase(currentStatus)) {
                        System.out.printf("\r Neptune cluster reached desired status
'%s' after %s.\n", desiredStatus, formatElapsedTime((int) elapsedSeconds));
                        future.complete(null);
                    } else {
                        CompletableFuture.delayedExecutor(20, TimeUnit.SECONDS)
                                .execute(() ->
checkClusterStatusRecursive(clusterId, desiredStatus, startTime, future));
                    }
                });
    }


    /**
     * Starts an Amazon Neptune DB cluster.
     *
     * @param clusterIdentifier the unique identifier of the DB cluster to be
stopped
     */
    public CompletableFuture<StartDbClusterResponse> startDBClusterAsync(String
clusterIdentifier) {
        StartDbClusterRequest clusterRequest = StartDbClusterRequest.builder()
                .dbClusterIdentifier(clusterIdentifier)
                .build();

        return getAsyncClient().startDBCluster(clusterRequest)
                .whenComplete((response, error) -> {
                    if (error != null) {
                        Throwable cause = error.getCause() != null ?
error.getCause() : error;

                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }
```

```
                        throw new RuntimeException("Failed to start DB cluster: " +
cause.getMessage(), cause);
                    } else {
                        logger.info("DB Cluster starting: " + clusterIdentifier);
                    }
                });
    }

    /**
     * Stops an Amazon Neptune DB cluster.
     *
     * @param clusterIdentifier the unique identifier of the DB cluster to be
stopped
     */
    public CompletableFuture<StopDbClusterResponse> stopDBClusterAsync(String
clusterIdentifier) {
        StopDbClusterRequest clusterRequest = StopDbClusterRequest.builder()
                .dbClusterIdentifier(clusterIdentifier)
                .build();

        return getAsyncClient().stopDBCluster(clusterRequest)
                .whenComplete((response, error) -> {
                    if (error != null) {
                        Throwable cause = error.getCause() != null ?
error.getCause() : error;

                        if (cause instanceof ResourceNotFoundException) {
                            throw (ResourceNotFoundException) cause;
                        }

                        throw new RuntimeException("Failed to stop DB cluster: " +
cause.getMessage(), cause);
                    } else {
                        logger.info("DB Cluster stopped: " + clusterIdentifier);
                    }
                });
    }



    /**
     * Asynchronously describes the specified Amazon RDS DB cluster.
     *
```

```
    * @param clusterId the identifier of the DB cluster to describe
    * @return a {@link CompletableFuture} that completes when the operation is
done, or throws a {@link RuntimeException}
    * if an error occurs
    */
   public CompletableFuture<Void> describeDBClustersAsync(String clusterId) {
       DescribeDbClustersRequest request = DescribeDbClustersRequest.builder()
               .dbClusterIdentifier(clusterId)
               .build();

       return getAsyncClient().describeDBClusters(request)
               .thenAccept(response -> {
                   for (DBCluster cluster : response.dbClusters()) {
                       logger.info("Cluster Identifier: " +
cluster.dbClusterIdentifier());
                       logger.info("Status: " + cluster.status());
                       logger.info("Engine: " + cluster.engine());
                       logger.info("Engine Version: " + cluster.engineVersion());
                       logger.info("Endpoint: " + cluster.endpoint());
                       logger.info("Reader Endpoint: " + cluster.readerEndpoint());
                       logger.info("Availability Zones: " +
cluster.availabilityZones());
                       logger.info("Subnet Group: " + cluster.dbSubnetGroup());
                       logger.info("VPC Security Groups:");
                       cluster.vpcSecurityGroups().forEach(vpcGroup ->
                               logger.info("  - " +
vpcGroup.vpcSecurityGroupId()));
                       logger.info("Storage Encrypted: " +
cluster.storageEncrypted());
                       logger.info("IAM DB Auth Enabled: " +
cluster.iamDatabaseAuthenticationEnabled());
                       logger.info("Backup Retention Period: " +
cluster.backupRetentionPeriod() + " days");
                       logger.info("Preferred Backup Window: " +
cluster.preferredBackupWindow());
                       logger.info("Preferred Maintenance Window: " +
cluster.preferredMaintenanceWindow());
                       logger.info("------");
                   }
               })
               .exceptionally(ex -> {
                   Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                   if (cause instanceof ResourceNotFoundException) {
```

```
                        throw (ResourceNotFoundException) cause;
                    }

                    throw new RuntimeException("Failed to describe the DB cluster: "
+ cause.getMessage(), cause);
                });
    }


    public CompletableFuture<Void> checkInstanceStatus(String instanceId, String
desiredStatus) {
        CompletableFuture<Void> future = new CompletableFuture<>();
        long startTime = System.currentTimeMillis();
        checkStatusRecursive(instanceId, desiredStatus.toLowerCase(), startTime,
future);
        return future;
    }

    /**
     * Checks the status of a Neptune instance recursively until the desired status
is reached or a timeout occurs.
     *
     * @param instanceId     the ID of the Neptune instance to check
     * @param desiredStatus  the desired status of the Neptune instance
     * @param startTime      the start time of the operation, used to calculate the
elapsed time
     * @param future         a {@link CompletableFuture} that will be completed when
the desired status is reached
     */
    private void checkStatusRecursive(String instanceId, String desiredStatus, long
startTime, CompletableFuture<Void> future) {
        DescribeDbInstancesRequest request = DescribeDbInstancesRequest.builder()
                .dbInstanceIdentifier(instanceId)
                .build();

        getAsyncClient().describeDBInstances(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        future.completeExceptionally(
                                new CompletionException("Error checking Neptune
instance status", cause)
                        );
                        return;
```

```
                    }

                    List<DBInstance> instances = response.dbInstances();
                    if (instances.isEmpty()) {
                        future.completeExceptionally(new RuntimeException("Instance
not found: " + instanceId));
                        return;
                    }

                    String currentStatus = instances.get(0).dbInstanceStatus();
                    long elapsedSeconds = (System.currentTimeMillis() - startTime) /
1000;
                    System.out.printf("\r Elapsed: %-20s  Status: %-20s",
formatElapsedTime((int) elapsedSeconds), currentStatus);
                    System.out.flush();

                    if (desiredStatus.equalsIgnoreCase(currentStatus)) {
                        System.out.printf("\r Neptune instance reached desired
status '%s' after %s.\n", desiredStatus, formatElapsedTime((int) elapsedSeconds));
                        future.complete(null);
                    } else {
                        CompletableFuture.delayedExecutor(20, TimeUnit.SECONDS)
                                .execute(() -> checkStatusRecursive(instanceId,
desiredStatus, startTime, future));
                    }
                });
    }


    private String formatElapsedTime(int seconds) {
        int minutes = seconds / 60;
        int remainingSeconds = seconds % 60;

        if (minutes > 0) {
            return minutes + (minutes == 1 ? " min" : " mins") + ", " +
                    remainingSeconds + (remainingSeconds == 1 ? " sec" : " secs");
        } else {
            return remainingSeconds + (remainingSeconds == 1 ? " sec" : " secs");
        }
    }


    /**
     * Creates a new Amazon Neptune DB instance asynchronously.
```

```
    *
    * @param dbInstanceId the identifier for the new DB instance
    * @param dbClusterId  the identifier for the DB cluster that the new instance
will be a part of
    * @return a {@link CompletableFuture} that completes with the identifier of the
newly created DB instance
    * @throws CompletionException if the operation fails, with a cause of either:
    *                             - {@link ServiceQuotaExceededException} if the
request would exceed the maximum quota, or
    *                             - a general exception with the failure message
    */
   public CompletableFuture<String> createDBInstanceAsync(String dbInstanceId,
String dbClusterId) {
       CreateDbInstanceRequest request = CreateDbInstanceRequest.builder()
               .dbInstanceIdentifier(dbInstanceId)
               .dbInstanceClass("db.r5.large")
               .engine("neptune")
               .dbClusterIdentifier(dbClusterId)
               .build();

       return getAsyncClient().createDBInstance(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof ServiceQuotaExceededException) {
                           throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                       }
                       throw new CompletionException("Failed to create Neptune DB
instance: " + exception.getMessage(), exception);
                   }
               })
               .thenApply(response -> {
                   String instanceId =
response.dbInstance().dbInstanceIdentifier();
                   logger.info("Created Neptune DB Instance: " + instanceId);
                   return instanceId;
               });
   }


   /**
    * Creates a new Amazon Neptune DB cluster asynchronously.
    *
```

```
    * @param dbName the name of the DB cluster to be created
    * @return a CompletableFuture that, when completed, provides the ID of the
created DB cluster
    * @throws CompletionException if the operation fails for any reason, including
if the request would exceed the maximum quota
    */
   public CompletableFuture<String> createDBClusterAsync(String dbName) {
       CreateDbClusterRequest request = CreateDbClusterRequest.builder()
               .dbClusterIdentifier(dbName)
               .engine("neptune")
               .deletionProtection(false)
               .backupRetentionPeriod(1)
               .build();

       return getAsyncClient().createDBCluster(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof ServiceQuotaExceededException) {
                           throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                       }
                       throw new CompletionException("Failed to create Neptune DB
cluster: " + exception.getMessage(), exception);
                   }
               })
               .thenApply(response -> {
                   String clusterId = response.dbCluster().dbClusterIdentifier();
                   logger.info("DB Cluster created: " + clusterId);
                   return clusterId;
               });
   }


   /**
    * Creates a new DB subnet group asynchronously.
    *
    * @param groupName the name of the subnet group to create
    * @return a CompletableFuture that, when completed, returns the Amazon Resource
Name (ARN) of the created subnet group
    * @throws CompletionException if the operation fails, with a cause that may be
a ServiceQuotaExceededException if the request would exceed the maximum quota
    */
   public CompletableFuture<String> createSubnetGroupAsync(String groupName) {
```

```
        // Get the Amazon Virtual Private Cloud (VPC) where the Neptune cluster and
resources will be created
        String vpcId = getDefaultVpcId();
        logger.info("VPC is : " + vpcId);

        List<String> subnetList = getSubnetIds(vpcId);
        for (String subnetId : subnetList) {
            System.out.println("Subnet group:" +subnetId);
        }

        CreateDbSubnetGroupRequest request = CreateDbSubnetGroupRequest.builder()
                .dbSubnetGroupName(groupName)
                .dbSubnetGroupDescription("Subnet group for Neptune cluster")
                .subnetIds(subnetList)
                .build();

        return getAsyncClient().createDBSubnetGroup(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ServiceQuotaExceededException) {
                            throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                        }
                        throw new CompletionException("Failed to create subnet
group: " + exception.getMessage(), exception);
                    }
                })
                .thenApply(response -> {
                    String name = response.dbSubnetGroup().dbSubnetGroupName();
                    String arn = response.dbSubnetGroup().dbSubnetGroupArn();
                    logger.info("Subnet group created: " + name);
                    return arn;
                });
    }

    private List<String> getSubnetIds(String vpcId) {
        try (Ec2Client ec2 = Ec2Client.builder().region(region).build()) {
            DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
                    .filters(builder -> builder.name("vpc-id").values(vpcId))
                    .build();

            DescribeSubnetsResponse response = ec2.describeSubnets(request);
```

```
            return response.subnets().stream()
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());
        }
    }

    public static String getDefaultVpcId() {
        Ec2Client ec2 = Ec2Client.builder()
                .region(Region.US_EAST_1)
                .build();

        Filter myFilter = Filter.builder()
                .name("isDefault")
                .values("true")
                .build();

        List<Filter> filterList = new ArrayList<>();
        filterList.add(myFilter);

        DescribeVpcsRequest request = DescribeVpcsRequest.builder()
                .filters(filterList)
                .build();


        DescribeVpcsResponse response = ec2.describeVpcs(request);
        if (!response.vpcs().isEmpty()) {
            Vpc defaultVpc = response.vpcs().get(0);
            return defaultVpc.vpcId();
        } else {
            throw new RuntimeException("No default VPC found in this region.");
        }
    }
}
```

操作

CreateDBCluster

以下代码示例演示了如何使用 CreateDBCluster。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Creates a new Amazon Neptune DB cluster asynchronously.
    *
    * @param dbName the name of the DB cluster to be created
    * @return a CompletableFuture that, when completed, provides the ID of the
created DB cluster
    * @throws CompletionException if the operation fails for any reason, including
if the request would exceed the maximum quota
    */
   public CompletableFuture<String> createDBClusterAsync(String dbName) {
       CreateDbClusterRequest request = CreateDbClusterRequest.builder()
               .dbClusterIdentifier(dbName)
               .engine("neptune")
               .deletionProtection(false)
               .backupRetentionPeriod(1)
               .build();

       return getAsyncClient().createDBCluster(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof ServiceQuotaExceededException) {
                           throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                       }
                       throw new CompletionException("Failed to create Neptune DB
cluster: " + exception.getMessage(), exception);
                   }
               })
               .thenApply(response -> {
                   String clusterId = response.dbCluster().dbClusterIdentifier();
                   logger.info("DB Cluster created: " + clusterId);
                   return clusterId;
```

```
                });
    }
```

- 有关 API 的详细信息，请参阅DBCluster在 AWS SDK for Java 2.x API 参考中创建。

`CreateDBInstance`

以下代码示例演示了如何使用 `CreateDBInstance`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Creates a new Amazon Neptune DB instance asynchronously.
     *
     * @param dbInstanceId the identifier for the new DB instance
     * @param dbClusterId  the identifier for the DB cluster that the new instance
will be a part of
     * @return a {@link CompletableFuture} that completes with the identifier of the
newly created DB instance
     * @throws CompletionException if the operation fails, with a cause of either:
     *                             - {@link ServiceQuotaExceededException} if the
request would exceed the maximum quota, or
     *                             - a general exception with the failure message
     */
    public CompletableFuture<String> createDBInstanceAsync(String dbInstanceId,
String dbClusterId) {
        CreateDbInstanceRequest request = CreateDbInstanceRequest.builder()
                .dbInstanceIdentifier(dbInstanceId)
                .dbInstanceClass("db.r5.large")
                .engine("neptune")
                .dbClusterIdentifier(dbClusterId)
                .build();

        return getAsyncClient().createDBInstance(request)
```

```
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        if (cause instanceof ServiceQuotaExceededException) {
                            throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                        }
                        throw new CompletionException("Failed to create Neptune DB
instance: " + exception.getMessage(), exception);
                    }
                })
                .thenApply(response -> {
                    String instanceId =
response.dbInstance().dbInstanceIdentifier();
                    logger.info("Created Neptune DB Instance: " + instanceId);
                    return instanceId;
                });
    }
```

- 有关 API 的详细信息，请参阅DBInstance在 AWS SDK for Java 2.x API 参考中创建。

`CreateDBSubnetGroup`

以下代码示例演示了如何使用 `CreateDBSubnetGroup`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Creates a new DB subnet group asynchronously.
     *
     * @param groupName the name of the subnet group to create
     * @return a CompletableFuture that, when completed, returns the Amazon Resource
Name (ARN) of the created subnet group
```

```
    * @throws CompletionException if the operation fails, with a cause that may be
a ServiceQuotaExceededException if the request would exceed the maximum quota
    */
   public CompletableFuture<String> createSubnetGroupAsync(String groupName) {

       // Get the Amazon Virtual Private Cloud (VPC) where the Neptune cluster and
resources will be created
       String vpcId = getDefaultVpcId();
       logger.info("VPC is : " + vpcId);

       List<String> subnetList = getSubnetIds(vpcId);
       for (String subnetId : subnetList) {
           System.out.println("Subnet group:" +subnetId);
       }

       CreateDbSubnetGroupRequest request = CreateDbSubnetGroupRequest.builder()
               .dbSubnetGroupName(groupName)
               .dbSubnetGroupDescription("Subnet group for Neptune cluster")
               .subnetIds(subnetList)
               .build();

       return getAsyncClient().createDBSubnetGroup(request)
               .whenComplete((response, exception) -> {
                   if (exception != null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof ServiceQuotaExceededException) {
                           throw new CompletionException("The operation was denied
because the request would exceed the maximum quota.", cause);
                       }
                       throw new CompletionException("Failed to create subnet
group: " + exception.getMessage(), exception);
                   }
               })
               .thenApply(response -> {
                   String name = response.dbSubnetGroup().dbSubnetGroupName();
                   String arn = response.dbSubnetGroup().dbSubnetGroupArn();
                   logger.info("Subnet group created: " + name);
                   return arn;
               });
   }
```

- 有关 API 的详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 “创建DBSubnet群组”。

CreateGraph

以下代码示例演示了如何使用 CreateGraph。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Executes the process of creating a new Neptune graph.
    *
    * @param client        the Neptune graph client used to interact with the
Neptune service
    * @param graphName     the name of the graph to be created
    * @throws NeptuneGraphException if an error occurs while creating the graph
    */
   public static void executeCreateGraph(NeptuneGraphClient client, String
graphName) {
       try {
           // Create the graph request
           CreateGraphRequest request = CreateGraphRequest.builder()
                   .graphName(graphName)
                   .provisionedMemory(16)
                   .build();

           // Create the graph
           CreateGraphResponse response = client.createGraph(request);

           // Extract the graph name and ARN
           String createdGraphName = response.name();
           String graphArn = response.arn();
           String graphEndpoint = response.endpoint();

           System.out.println("Graph created successfully!");
           System.out.println("Graph Name: " + createdGraphName);
           System.out.println("Graph ARN: " + graphArn);
           System.out.println("Graph Endpoint: " +graphEndpoint );

       } catch (NeptuneGraphException e) {
```

```
            System.err.println("Failed to create graph: " +
e.awsErrorDetails().errorMessage());
        } finally {
            client.close();
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateGraph中的。

`DeleteDBCluster`

以下代码示例演示了如何使用 `DeleteDBCluster`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Deletes a DB instance asynchronously.
     *
     * @param clusterId the identifier of the cluster to delete
     * @return a {@link CompletableFuture} that completes when the cluster has been
deleted
     */
    public CompletableFuture<Void> deleteDBClusterAsync(String clusterId) {
        DeleteDbClusterRequest request = DeleteDbClusterRequest.builder()
                .dbClusterIdentifier(clusterId)
                .skipFinalSnapshot(true)
                .build();

        return getAsyncClient().deleteDBCluster(request)
                .thenAccept(response -> System.out.println("## Deleting DB Cluster:
" + clusterId));
    }
```

- 有关 API 的详细信息，请参阅DBCluster《AWS SDK for Java 2.x API 参考》中的 “删除”。

DeleteDBInstance

以下代码示例演示了如何使用 DeleteDBInstance。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Deletes a DB instance asynchronously.
     *
     * @param instanceId the identifier of the DB instance to be deleted
     * @return a {@link CompletableFuture} that completes when the DB instance has
been deleted
     */
    public CompletableFuture<Void> deleteDBInstanceAsync(String instanceId) {
        DeleteDbInstanceRequest request = DeleteDbInstanceRequest.builder()
                .dbInstanceIdentifier(instanceId)
                .skipFinalSnapshot(true)
                .build();

        return getAsyncClient().deleteDBInstance(request)
                .thenAccept(response -> System.out.println("## Deleting DB Instance:
" + instanceId));
    }
```

- 有关 API 的详细信息，请参阅DBInstance《AWS SDK for Java 2.x API 参考》中的 “删除”。

DeleteDBSubnetGroup

以下代码示例演示了如何使用 DeleteDBSubnetGroup。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Deletes a subnet group.
     *
     * @param subnetGroupName the identifier of the subnet group to delete
     * @return a {@link CompletableFuture} that completes when the cluster has been
deleted
     */
    public CompletableFuture<Void> deleteDBSubnetGroupAsync(String subnetGroupName)
{
        DeleteDbSubnetGroupRequest request = DeleteDbSubnetGroupRequest.builder()
                .dbSubnetGroupName(subnetGroupName)
                .build();

        return getAsyncClient().deleteDBSubnetGroup(request)
                .thenAccept(response -> logger.info("## Deleting Subnet Group: " +
subnetGroupName));
    }
```

- 有关 API 的详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 “删除DBSubnet群组”。

`DescribeDBClusters`

以下代码示例演示了如何使用 `DescribeDBClusters`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Asynchronously describes the specified Amazon RDS DB cluster.
    *
    * @param clusterId the identifier of the DB cluster to describe
    * @return a {@link CompletableFuture} that completes when the operation is
done, or throws a {@link RuntimeException}
    * if an error occurs
    */
   public CompletableFuture<Void> describeDBClustersAsync(String clusterId) {
       DescribeDbClustersRequest request = DescribeDbClustersRequest.builder()
               .dbClusterIdentifier(clusterId)
               .build();

       return getAsyncClient().describeDBClusters(request)
               .thenAccept(response -> {
                   for (DBCluster cluster : response.dbClusters()) {
                       logger.info("Cluster Identifier: " +
cluster.dbClusterIdentifier());
                       logger.info("Status: " + cluster.status());
                       logger.info("Engine: " + cluster.engine());
                       logger.info("Engine Version: " + cluster.engineVersion());
                       logger.info("Endpoint: " + cluster.endpoint());
                       logger.info("Reader Endpoint: " + cluster.readerEndpoint());
                       logger.info("Availability Zones: " +
cluster.availabilityZones());
                       logger.info("Subnet Group: " + cluster.dbSubnetGroup());
                       logger.info("VPC Security Groups:");
                       cluster.vpcSecurityGroups().forEach(vpcGroup ->
                               logger.info("  - " +
vpcGroup.vpcSecurityGroupId()));
                       logger.info("Storage Encrypted: " +
cluster.storageEncrypted());
                       logger.info("IAM DB Auth Enabled: " +
cluster.iamDatabaseAuthenticationEnabled());
                       logger.info("Backup Retention Period: " +
cluster.backupRetentionPeriod() + " days");
                       logger.info("Preferred Backup Window: " +
cluster.preferredBackupWindow());
                       logger.info("Preferred Maintenance Window: " +
cluster.preferredMaintenanceWindow());
                       logger.info("------");
                   }
```

```
                })
                .exceptionally(ex -> {
                    Throwable cause = ex.getCause() != null ? ex.getCause() : ex;

                    if (cause instanceof ResourceNotFoundException) {
                        throw (ResourceNotFoundException) cause;
                    }

                    throw new RuntimeException("Failed to describe the DB cluster: "
+ cause.getMessage(), cause);
                });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DBClusters中的描述。

`DescribeDBInstances`

以下代码示例演示了如何使用 `DescribeDBInstances`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Checks the status of a Neptune instance recursively until the desired status
is reached or a timeout occurs.
     *
     * @param instanceId     the ID of the Neptune instance to check
     * @param desiredStatus  the desired status of the Neptune instance
     * @param startTime      the start time of the operation, used to calculate the
elapsed time
     * @param future         a {@link CompletableFuture} that will be completed when
the desired status is reached
     */
    private void checkStatusRecursive(String instanceId, String desiredStatus, long
startTime, CompletableFuture<Void> future) {
```

```
        DescribeDbInstancesRequest request = DescribeDbInstancesRequest.builder()
                .dbInstanceIdentifier(instanceId)
                .build();

        getAsyncClient().describeDBInstances(request)
                .whenComplete((response, exception) -> {
                    if (exception != null) {
                        Throwable cause = exception.getCause();
                        future.completeExceptionally(
                                new CompletionException("Error checking Neptune
instance status", cause)
                        );
                        return;
                    }

                    List<DBInstance> instances = response.dbInstances();
                    if (instances.isEmpty()) {
                        future.completeExceptionally(new RuntimeException("Instance
not found: " + instanceId));
                        return;
                    }

                    String currentStatus = instances.get(0).dbInstanceStatus();
                    long elapsedSeconds = (System.currentTimeMillis() - startTime) /
1000;
                    System.out.printf("\r Elapsed: %-20s  Status: %-20s",
formatElapsedTime((int) elapsedSeconds), currentStatus);
                    System.out.flush();

                    if (desiredStatus.equalsIgnoreCase(currentStatus)) {
                        System.out.printf("\r Neptune instance reached desired
status '%s' after %s.\n", desiredStatus, formatElapsedTime((int) elapsedSeconds));
                        future.complete(null);
                    } else {
                        CompletableFuture.delayedExecutor(20, TimeUnit.SECONDS)
                                .execute(() -> checkStatusRecursive(instanceId,
desiredStatus, startTime, future));
                    }
                });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DBInstances中的描述。

ExecuteGremlinProfileQuery

以下代码示例演示了如何使用 ExecuteGremlinProfileQuery。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Executes a Gremlin query against an Amazon Neptune database using the
provided {@link NeptunedataClient}.
    *
    * @param client the {@link NeptunedataClient} instance to use for executing the
Gremlin query
    */
   public static void executeGremlinQuery(NeptunedataClient client) {
       try {
           System.out.println("Querying Neptune...");
           ExecuteGremlinQueryRequest request =
ExecuteGremlinQueryRequest.builder()
                   .gremlinQuery("g.V().has('code', 'ANC')")
                   .build();

           ExecuteGremlinQueryResponse response =
client.executeGremlinQuery(request);

           System.out.println("Full Response:");
           System.out.println(response);

           // Retrieve and print the result
           if (response.result() != null) {
               System.out.println("Query Result:");
               System.out.println(response.result().toString());
           } else {
               System.out.println("No result returned from the query.");
           }
       } catch (NeptunedataException e) {
           System.err.println("Error calling Neptune: " +
e.awsErrorDetails().errorMessage());
```

```
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
        } finally {
            client.close();
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ExecuteGremlinProfileQuery](#)中的。

`ExecuteGremlinQuery`

以下代码示例演示了如何使用 `ExecuteGremlinQuery`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Executes a Gremlin PROFILE query using the provided NeptunedataClient.
     *
     * @param client The NeptunedataClient instance to be used for executing the
Gremlin PROFILE query.
     */
    private static void executeGremlinProfileQuery(NeptunedataClient client) {
        System.out.println("Executing Gremlin PROFILE query...");

        ExecuteGremlinProfileQueryRequest request =
ExecuteGremlinProfileQueryRequest.builder()
                .gremlinQuery("g.V().has('code', 'ANC')")
                .build();

        ExecuteGremlinProfileQueryResponse response =
client.executeGremlinProfileQuery(request);
        if (response.output() != null) {
            System.out.println("Query Profile Output:");
```

```
            System.out.println(response.output());
        } else {
            System.out.println("No output returned from the profile query.");
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ExecuteGremlinQuery](#)中的。

`ExecuteOpenCypherExplainQuery`

以下代码示例演示了如何使用 `ExecuteOpenCypherExplainQuery`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Executes an OpenCypher EXPLAIN query using the provided Neptune data client.
     *
     * @param client The Neptune data client to use for the query execution.
     */
    public static void executeGremlinQuery(NeptunedataClient client) {
        try {
            System.out.println("Executing OpenCypher EXPLAIN query...");
            ExecuteOpenCypherExplainQueryRequest request =
ExecuteOpenCypherExplainQueryRequest.builder()
                    .openCypherQuery("MATCH (n {code: 'ANC'}) RETURN n")
                    .explainMode("debug")
                    .build();

            ExecuteOpenCypherExplainQueryResponse response =
client.executeOpenCypherExplainQuery(request);

            if (response.results() != null) {
                System.out.println("Explain Results:");
                System.out.println(response.results().asUtf8String());
```

```
            } else {
                System.out.println("No explain results returned.");
            }

        } catch (NeptunedataException e) {
            System.err.println("Neptune error: " +
e.awsErrorDetails().errorMessage());
        } catch (Exception e) {
            System.err.println("Unexpected error: " + e.getMessage());
        } finally {
            client.close();
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ExecuteOpenCypherExplainQuery](#)中的。

`ExecuteQuery`

以下代码示例演示了如何使用 `ExecuteQuery`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Executes a Gremlin profile query on the Neptune Analytics graph.
     *
     * @param client       the {@link NeptuneGraphClient} instance to use for the
query
     * @param graphId      the identifier of the graph to execute the query on
     *
     * @throws NeptuneGraphException if an error occurs while executing the query on
the Neptune Graph
     * @throws Exception if an unexpected error occurs
     */
```

```
   public static void executeGremlinProfileQuery(NeptuneGraphClient client, String
graphId) {

       try {
           System.out.println("Running openCypher query on Neptune Analytics...");

           ExecuteQueryRequest request = ExecuteQueryRequest.builder()
                   .graphIdentifier(graphId)
                   .queryString("MATCH (n {code: 'ANC'}) RETURN n")
                   .language("OPEN_CYPHER")
                   .build();

           ResponseInputStream<ExecuteQueryResponse> response =
client.executeQuery(request);
           try (BufferedReader reader = new BufferedReader(new
InputStreamReader(response, StandardCharsets.UTF_8))) {
               String result = reader.lines().collect(Collectors.joining("\n"));
               System.out.println("Query Result:");
               System.out.println(result);
           } catch (Exception e) {
               System.err.println("Error reading response: " + e.getMessage());
           }

       } catch (NeptuneGraphException e) {
           System.err.println("NeptuneGraph error: " +
e.awsErrorDetails().errorMessage());
       } catch (Exception e) {
           System.err.println("Unexpected error: " + e.getMessage());
       } finally {
           client.close();
       }
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ExecuteQuery中的。

StartDBCluster

以下代码示例演示了如何使用 StartDBCluster。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Starts an Amazon Neptune DB cluster.
    *
    * @param clusterIdentifier the unique identifier of the DB cluster to be
stopped
    */
   public CompletableFuture<StartDbClusterResponse> startDBClusterAsync(String
clusterIdentifier) {
       StartDbClusterRequest clusterRequest = StartDbClusterRequest.builder()
               .dbClusterIdentifier(clusterIdentifier)
               .build();

       return getAsyncClient().startDBCluster(clusterRequest)
               .whenComplete((response, error) -> {
                   if (error != null) {
                       Throwable cause = error.getCause() != null ?
error.getCause() : error;

                       if (cause instanceof ResourceNotFoundException) {
                           throw (ResourceNotFoundException) cause;
                       }

                       throw new RuntimeException("Failed to start DB cluster: " +
cause.getMessage(), cause);
                   } else {
                       logger.info("DB Cluster starting: " + clusterIdentifier);
                   }
               });
   }
```

- 有关 API 的详细信息，请参阅DBCluster《AWS SDK for Java 2.x API 参考》中的 “开始”。

StopDBCluster

以下代码示例演示了如何使用 StopDBCluster。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * Stops an Amazon Neptune DB cluster.
    *
    * @param clusterIdentifier the unique identifier of the DB cluster to be
stopped
    */
   public CompletableFuture<StopDbClusterResponse> stopDBClusterAsync(String
clusterIdentifier) {
       StopDbClusterRequest clusterRequest = StopDbClusterRequest.builder()
               .dbClusterIdentifier(clusterIdentifier)
               .build();

       return getAsyncClient().stopDBCluster(clusterRequest)
               .whenComplete((response, error) -> {
                   if (error != null) {
                       Throwable cause = error.getCause() != null ?
error.getCause() : error;

                       if (cause instanceof ResourceNotFoundException) {
                           throw (ResourceNotFoundException) cause;
                       }

                       throw new RuntimeException("Failed to stop DB cluster: " +
cause.getMessage(), cause);
                   } else {
                       logger.info("DB Cluster stopped: " + clusterIdentifier);
                   }
               });
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DBCluster中的停止。

场景

使用 Neptune API 查询图形数据

以下代码示例展示了如何使用 Neptune API 查询图形数据。

适用于 Java 的 SDK 2.x

展示如何使用 Amazon Neptune Java API 创建查询 VPC 内的图形数据的 Lambda 函数。

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

- Lambda
- Neptune

使用适用于 Java 的 SDK 2.x 的合作伙伴中心示例

以下代码示例向您展示了如何使用与 Partner Central AWS SDK for Java 2.x 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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从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

AssignOpportunity

以下代码示例演示了如何使用 AssignOpportunity。

适用于 Java 的 SDK 2.x

将现有机会重新分配给其他用户。

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssignOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssignOpportunityResponse;
import software.amazon.awssdk.services.partnercentralselling.model.AssigneeContact;

/*
Purpose
PC-API-07 Assigning a new owner
*/

public class AssignOpportunity {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {
```

```
    String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

    String assigneeFirstName = "John";

    String assigneeLastName = "Doe";

    String assigneeEmail = "test@test.com";

    String businessTitle = "PartnerAccountManager";

    AssignOpportunityResponse response = getResponse(opportunityId,
 assigneeFirstName, assigneeLastName, assigneeEmail, businessTitle);

    ReferenceCodesUtils.formatOutput(response);
   }

 static AssignOpportunityResponse getResponse(String opportunityId, String
 assigneeFirstName, String assigneeLastName, String assigneeEmail, String
 businessTitle) {

  AssignOpportunityRequest assignOpportunityRequest =
 AssignOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
          .assignee(AssigneeContact.builder()
            .firstName(assigneeFirstName)
            .lastName(assigneeLastName)
            .email(assigneeEmail)
            .businessTitle(businessTitle)
            .build())
          .build();

       AssignOpportunityResponse response =
 client.assignOpportunity(assignOpportunityRequest);

       return response;
 }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[AssignOpportunity](#)中的。

AssociateOpportunity

以下代码示例演示了如何使用 AssociateOpportunity。

适用于 Java 的 SDK 2.x

在机会与各种相关实体之间创建正式关联。

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityResponse;

/*
Purpose
PC-API -11 Associating a product
PC-API -12 Associating a solution
PC-API -13 Associating an offer
entity_type = Solutions | AWSProducts | AWSMarketplaceOffers
*/

public class AssociateOpportunity {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;
```

```
     String entityType = "Solutions";

     String entityIdentifier = "S-0000000";

     AssociateOpportunityResponse response = getResponse(opportunityId, entityType,
 entityIdentifier );

     ReferenceCodesUtils.formatOutput(response);
    }

static AssociateOpportunityResponse getResponse(String opportunityId, String
 entityType, String entityIdentifier) {

        AssociateOpportunityRequest associateOpportunityRequest =
 AssociateOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .opportunityIdentifier(opportunityId)
          .relatedEntityType(entityType)
          .relatedEntityIdentifier(entityIdentifier)
          .build();

        AssociateOpportunityResponse response =
 client.associateOpportunity(associateOpportunityRequest);

        return response;
 }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[AssociateOpportunity](#)中的。

`CreateOpportunity`

以下代码示例演示了如何使用 `CreateOpportunity`。

适用于 Java 的 SDK 2.x

创建机会。

```
package org.example;

import java.time.Instant;
```

```
import java.util.ArrayList;
import java.util.List;

import static org.example.utils.Constants.*;

import org.example.entity.Root;
import org.example.utils.ReferenceCodesUtils;
import org.example.utils.StringSerializer;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import software.amazon.awssdk.services.partnercentralselling.model.Account;
import software.amazon.awssdk.services.partnercentralselling.model.Address;
import software.amazon.awssdk.services.partnercentralselling.model.Contact;
import
 software.amazon.awssdk.services.partnercentralselling.model.CreateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.CreateOpportunityResponse;
import software.amazon.awssdk.services.partnercentralselling.model.Customer;
import
 software.amazon.awssdk.services.partnercentralselling.model.ExpectedCustomerSpend;
import software.amazon.awssdk.services.partnercentralselling.model.LifeCycle;
import software.amazon.awssdk.services.partnercentralselling.model.Marketing;
import software.amazon.awssdk.services.partnercentralselling.model.MonetaryValue;
import software.amazon.awssdk.services.partnercentralselling.model.NextStepsHistory;
import software.amazon.awssdk.services.partnercentralselling.model.Project;
import software.amazon.awssdk.services.partnercentralselling.model.SoftwareRevenue;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.ToNumberPolicy;

public class CreateOpportunity {

 static final Gson GSON = new GsonBuilder()
   .setObjectToNumberStrategy(ToNumberPolicy.LAZILY_PARSED_NUMBER)
   .registerTypeAdapter(String.class, new StringSerializer())
   .create();

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
```

```
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

public static void main(String[] args) {

 String inputFile = "CreateOpportunity2.json";

 if (args.length > 0)
  inputFile = args[0];

 CreateOpportunityResponse response = createOpportunity(inputFile);

 client.close();
}

static CreateOpportunityResponse createOpportunity(String inputFile) {

 String inputString = ReferenceCodesUtils.readInputFileToString(inputFile);

 Root root = GSON.fromJson(inputString, Root.class);

 List<NextStepsHistory> nextStepsHistories = new ArrayList<NextStepsHistory>();
 if ( root.lifeCycle != null && root.lifeCycle.nextStepsHistories != null) {
  for (org.example.entity.NextStepsHistory nextStepsHistoryJson :
root.lifeCycle.nextStepsHistories) {
   NextStepsHistory nextStepsHistory = NextStepsHistory.builder()
     .time(Instant.parse(nextStepsHistoryJson.time))
     .value(nextStepsHistoryJson.value)
                 .build();
   nextStepsHistories.add(nextStepsHistory);
  }
 }

 LifeCycle lifeCycle = null;
 if ( root.lifeCycle != null ) {
  lifeCycle = LifeCycle.builder()
   .closedLostReason(root.lifeCycle.closedLostReason)
   .nextSteps(root.lifeCycle.nextSteps)
   .nextStepsHistory(nextStepsHistories)
   .reviewComments(root.lifeCycle.reviewComments)
   .reviewStatus(root.lifeCycle.reviewStatus)
   .reviewStatusReason(root.lifeCycle.reviewStatusReason)
   .stage(root.lifeCycle.stage)
```

```
    .targetCloseDate(root.lifeCycle.targetCloseDate)
    .build();
 }

 Marketing marketing = null;
 if ( root.marketing != null ) {
  marketing = Marketing.builder()
     .awsFundingUsed(root.marketing.awsFundingUsed)
     .campaignName(root.marketing.campaignName)
     .channels(root.marketing.channels)
     .source(root.marketing.source)
     .useCases(root.marketing.useCases)
     .build();

 }

 Address address = null;
 if ( root.customer != null && root.customer.account != null &&
root.customer.account.address != null ) {
  address = Address.builder()
   .city(root.customer.account.address.city)
               .postalCode(root.customer.account.address.postalCode)
               .stateOrRegion(root.customer.account.address.stateOrRegion)
               .countryCode(root.customer.account.address.countryCode)
               .streetAddress(root.customer.account.address.streetAddress)
               .build();
 }

 Account account = null;
 if ( root.customer != null && root.customer.account!= null) {
  account = Account.builder()
            .address(address)
            .awsAccountId(root.customer.account.awsAccountId)
               .duns(root.customer.account.duns)
               .industry(root.customer.account.industry)
               .otherIndustry(root.customer.account.otherIndustry)
               .companyName(root.customer.account.companyName)
               .websiteUrl(root.customer.account.websiteUrl)
               .build();
 }

 List<Contact> contacts = new ArrayList<Contact>();
 if ( root.customer != null && root.customer.contacts != null) {
  for (org.example.entity.Contact jsonContact : root.customer.contacts) {
```

```
    Contact contact = Contact.builder()
                  .email(jsonContact.email)
                  .firstName(jsonContact.firstName)
                  .lastName(jsonContact.lastName)
                  .phone(jsonContact.phone)
                  .businessTitle(jsonContact.businessTitle)
                  .build();
    contacts.add(contact);
   }
  }

  Customer customer = Customer.builder()
    .account(account)
    .contacts(contacts)
    .build();

  Contact oportunityTeamContact = null;
  if (root.opportunityTeam != null && root.opportunityTeam.get(0) != null ) {
   oportunityTeamContact = Contact.builder()
                .firstName(root.opportunityTeam.get(0).firstName)
                .lastName(root.opportunityTeam.get(0).lastName)
                .email(root.opportunityTeam.get(0).email)
                .phone(root.opportunityTeam.get(0).phone)
                .businessTitle(root.opportunityTeam.get(0).businessTitle)
                .build();
  }

  List<ExpectedCustomerSpend> expectedCustomerSpends = new
ArrayList<ExpectedCustomerSpend>();
  if ( root.project != null && root.project.expectedCustomerSpend != null) {
   for (org.example.entity.ExpectedCustomerSpend expectedCustomerSpendJson :
root.project.expectedCustomerSpend) {
    ExpectedCustomerSpend expectedCustomerSpend = null;
    expectedCustomerSpend = ExpectedCustomerSpend.builder()
      .amount(expectedCustomerSpendJson.amount)
      .currencyCode(expectedCustomerSpendJson.currencyCode)
      .frequency(expectedCustomerSpendJson.frequency)
      .targetCompany(expectedCustomerSpendJson.targetCompany)
      .build();
    expectedCustomerSpends.add(expectedCustomerSpend);
   }
        }

        Project project = null;
```

```
        if ( root.project != null) {
         project = Project.builder()
                .title(root.project.title)
                .customerBusinessProblem(root.project.customerBusinessProblem)
                .customerUseCase(root.project.customerUseCase)
                .deliveryModels(root.project.deliveryModels)
                .expectedCustomerSpend(expectedCustomerSpends)
                .salesActivities(root.project.salesActivities)
                .competitorName(root.project.competitorName)
                .otherSolutionDescription(root.project.otherSolutionDescription)
                .build();
        }

        SoftwareRevenue softwareRevenue = null;
        if ( root.softwareRevenue != null) {
         MonetaryValue monetaryValue = null;
         if ( root.softwareRevenue.value != null) {
          monetaryValue = MonetaryValue.builder()
             .amount(root.softwareRevenue.value.amount)
             .currencyCode(root.softwareRevenue.value.currencyCode)
             .build();
         }
         softwareRevenue = SoftwareRevenue.builder()
           .deliveryModel(root.softwareRevenue.deliveryModel)
           .effectiveDate(root.softwareRevenue.effectiveDate)
           .expirationDate(root.softwareRevenue.expirationDate)
           .value(monetaryValue)
           .build();
        }

 // Building the Actual CreateOpportunity Request
 CreateOpportunityRequest createOpportunityRequest =
CreateOpportunityRequest.builder()
   .catalog(CATALOG_TO_USE)
   .clientToken(root.clientToken)
   .primaryNeedsFromAwsWithStrings(root.primaryNeedsFromAws)
   .opportunityType(root.opportunityType)
   .lifeCycle(lifeCycle)
   .marketing(marketing)
   .nationalSecurity(root.nationalSecurity)
   .origin(root.origin)
   .customer(customer)
   .project(project)
   .partnerOpportunityIdentifier(root.partnerOpportunityIdentifier)
```

```
    .opportunityTeam(oportunityTeamContact)
    .softwareRevenue(softwareRevenue)
    .build();

  CreateOpportunityResponse response =
 client.createOpportunity(createOpportunityRequest);
  System.out.println("Successfully created: " + response);

  return response;
    }

}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateOpportunity中的。

`DisassociateOpportunity`

以下代码示例演示了如何使用 `DisassociateOpportunity`。

适用于 Java 的 SDK 2.x

移除机会与相关实体之间的现有关联。

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityResponse

/*
Purpose
```

```
PC-API -14 Removing a Solution
PC-API -15 Removing an offer
PC-API -16 Removing a product
entity_type = Solutions | AWSProducts | AWSMarketplaceOffers
*/

public class DisassociateOpportunity {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     String entityType = "Solutions";

     String entityIdentifier = "S-0000000";

     DisassociateOpportunityResponse response = getResponse(opportunityId,
 entityType, entityIdentifier );

     ReferenceCodesUtils.formatOutput(response);
    }

 static DisassociateOpportunityResponse getResponse(String opportunityId, String
 entityType, String entityIdentifier) {

  DisassociateOpportunityRequest disassociateOpportunityRequest =
 DisassociateOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .opportunityIdentifier(opportunityId)
          .relatedEntityType(entityType)
          .relatedEntityIdentifier(entityIdentifier)
          .build();

        DisassociateOpportunityResponse response =
 client.disassociateOpportunity(disassociateOpportunityRequest);

        return response;
 }
```

```
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DisassociateOpportunity中的。

GetAwsOpportunitySummary

以下代码示例演示了如何使用 GetAwsOpportunitySummary。

适用于 Java 的 SDK 2.x

检索 AWS 机会摘要。

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetAwsOpportunitySummaryRequest
import
 software.amazon.awssdk.services.partnercentralselling.model.GetAwsOpportunitySummaryRespons

/*
 * Purpose
 * PC-API-25 Retrieves a summary of an AWS Opportunity.
 */

public class GetAwsOpportunitySummary {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();
```

```
    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     GetAwsOpportunitySummaryResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 public static GetAwsOpportunitySummaryResponse getResponse(String opportunityId) {

  GetAwsOpportunitySummaryRequest getOpportunityRequest =
 GetAwsOpportunitySummaryRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .relatedOpportunityIdentifier(opportunityId)
          .build();

  GetAwsOpportunitySummaryResponse response =
 client.getAwsOpportunitySummary(getOpportunityRequest);

        return response;
 }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[GetAwsOpportunitySummary](#)中的。

`GetEngagementInvitation`

以下代码示例演示了如何使用 `GetEngagementInvitation`。

适用于 Java 的 SDK 2.x

检索与合作伙伴共享的参与邀请 AWS 的详细信息。

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;
```

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationResponse

/*
 * Purpose
 * PC-API-22 Get engagement invitation opportunity
 */

public class GetEngagementInvitation {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     GetEngagementInvitationResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 static GetEngagementInvitationResponse getResponse(String opportunityId) {

  GetEngagementInvitationRequest getOpportunityRequest =
 GetEngagementInvitationRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
          .build();

  GetEngagementInvitationResponse response =
 client.getEngagementInvitation(getOpportunityRequest);

        return response;
```

```
 }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetEngagementInvitation中的。

`GetOpportunity`

以下代码示例演示了如何使用 `GetOpportunity`。

适用于 Java 的 SDK 2.x

获得机会。

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityResponse;

/*
 * Purpose
 * PC-API-08 Get updated Opportunity
 */

public class GetOpportunity {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();
```

```
    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     GetOpportunityResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 public static GetOpportunityResponse getResponse(String opportunityId) {

        GetOpportunityRequest getOpportunityRequest =
 GetOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
          .build();

        GetOpportunityResponse response =
 client.getOpportunity(getOpportunityRequest);

        return response;
 }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetOpportunity中的。

`ListEngagementInvitations`

以下代码示例演示了如何使用 `ListEngagementInvitations`。

适用于 Java 的 SDK 2.x

检索发送给合作伙伴的互动邀请列表。

```
package org.example;

import java.util.ArrayList;
import java.util.List;

import org.example.utils.ReferenceCodesUtils;
import static org.example.utils.Constants.*;
```

```
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListEngagementInvitationsReques
import
 software.amazon.awssdk.services.partnercentralselling.model.ListEngagementInvitationsRespon
import software.amazon.awssdk.services.partnercentralselling.model.ParticipantType;
import
 software.amazon.awssdk.services.partnercentralselling.model.EngagementInvitationSummary;

public class ListEngagementInvitations {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     List<EngagementInvitationSummary> opportunitySummaries = getResponse();
        ReferenceCodesUtils.formatOutput(opportunitySummaries);
    }

    static List<EngagementInvitationSummary> getResponse() {

  List<EngagementInvitationSummary> opportunitySummaries = new
 ArrayList<EngagementInvitationSummary>();

  ListEngagementInvitationsRequest listOpportunityRequest =
 ListEngagementInvitationsRequest.builder()
                .catalog(CATALOG_TO_USE)
                .participantType(ParticipantType.RECEIVER)
          .maxResults(5)
          .build();

  ListEngagementInvitationsResponse response =
 client.listEngagementInvitations(listOpportunityRequest);

     opportunitySummaries.addAll(response.engagementInvitationSummaries());
```

```
    client.close();

       return opportunitySummaries;
 }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListEngagementInvitations中的。

`ListOpportunities`

以下代码示例演示了如何使用 `ListOpportunities`。

适用于 Java 的 SDK 2.x

列出机会。

```
package org.example;

import java.util.ArrayList;
import java.util.List;

import org.example.utils.ReferenceCodesUtils;
import static org.example.utils.Constants.*;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListOpportunitiesRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListOpportunitiesResponse;
import
 software.amazon.awssdk.services.partnercentralselling.model.OpportunitySummary;

/*
 * Purpose
 * PC-API-18 Getting list of Opportunities
 */
```

```
public class ListOpportunititesPaging {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {
     List<OpportunitySummary> opportunitySummaries = getResponse();
        ReferenceCodesUtils.formatOutput(opportunitySummaries);
    }

    private static List<OpportunitySummary> getResponse() {
     List<OpportunitySummary> opportunitySummaries = new
 ArrayList<OpportunitySummary>();

  ListOpportunitiesRequest listOpportunityRequest =
 ListOpportunitiesRequest.builder()
                .catalog(CATALOG_TO_USE)
          .maxResults(5)
          .build();

     ListOpportunitiesResponse response =
 client.listOpportunities(listOpportunityRequest);

     opportunitySummaries.addAll(response.opportunitySummaries());

     while (response.nextToken() != null && response.nextToken().length() > 0) {
      listOpportunityRequest = ListOpportunitiesRequest.builder()
                    .catalog(CATALOG_TO_USE)
              .maxResults(5)
              .nextToken(response.nextToken())
              .build();
      response = client.listOpportunities(listOpportunityRequest);
      opportunitySummaries.addAll(response.opportunitySummaries());
     }

     client.close();

        return opportunitySummaries;
 }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListOpportunities中的。

`ListSolutions`

以下代码示例演示了如何使用 `ListSolutions`。

适用于 Java 的 SDK 2.x

检索合作伙伴在合作中心上注册的合作伙伴解决方案的列表。

```
package org.example;

import java.util.ArrayList;
import java.util.List;

import static org.example.utils.Constants.*;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListSolutionsRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.ListSolutionsResponse;
import software.amazon.awssdk.services.partnercentralselling.model.SolutionBase;

/*
 * Purpose
 * PC-API-10 Getting list of solutions
 */

public class ListSolutions {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();
```

```
    public static void main(String[] args) {
     List<SolutionBase> solutionSummaries = getResponse();
        ReferenceCodesUtils.formatOutput(solutionSummaries);
    }

    static List<SolutionBase> getResponse() {
  List<SolutionBase> solutionSummaries = new ArrayList<SolutionBase>();

  ListSolutionsRequest listSolutionsRequest = ListSolutionsRequest.builder()
    .catalog(CATALOG_TO_USE)
          .maxResults(5)
          .build();

     ListSolutionsResponse response = client.listSolutions(listSolutionsRequest);

     solutionSummaries.addAll(response.solutionSummaries());

        return solutionSummaries;
 }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ListSolutions](#)中的。

`RejectEngagementInvitation`

以下代码示例演示了如何使用 `RejectEngagementInvitation`。

适用于 Java 的 SDK 2.x

拒绝 AWS 共享 EngagementInvitation 的内容。

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
```

```
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.RejectEngagementInvitationReque
import
 software.amazon.awssdk.services.partnercentralselling.model.RejectEngagementInvitationRespo

/*
 * Purpose
 * PC-API-05 AWS Originated(AO) rejection
 */

public class RejectEngagementInvitation {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

  RejectEngagementInvitationResponse response = getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 static RejectEngagementInvitationResponse getResponse(String invitationId) {

        RejectEngagementInvitationRequest rejectOpportunityRequest =
 RejectEngagementInvitationRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(invitationId)
          .rejectionReason("Unable to support")
          .build();

  RejectEngagementInvitationResponse response =
 client.rejectEngagementInvitation(rejectOpportunityRequest);

        return response;
 }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考RejectEngagementInvitation中的。

StartEngagementByAcceptingInvitationTask

以下代码示例演示了如何使用 StartEngagementByAcceptingInvitationTask。

适用于 Java 的 SDK 2.x

通过接受 a 开始订婚 EngagementInvitation。

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementByAcceptingInvit
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementByAcceptingInvit
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetEngagementInvitationResponse
import software.amazon.awssdk.services.partnercentralselling.model.InvitationStatus;

/*
Purpose
PC-API-04: Start Engagement By Accepting InvitationTask for AWS Originated(AO)
 opportunity
*/

public class StartEngagementByAcceptingInvitationTask {
```

```
static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
           .region(Region.US_EAST_1)
           .credentialsProvider(DefaultCredentialsProvider.create())
           .httpClient(ApacheHttpClient.builder().build())
           .build();

static String clientToken = "test-a30d161";

   public static void main(String[] args) {

    String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

    StartEngagementByAcceptingInvitationTaskResponse response =
getResponse(opportunityId);

    if ( response == null) {
     System.out.println("Opportunity is not AWS Originated.");
    } else {
     ReferenceCodesUtils.formatOutput(response);
    }
   }

   private static GetEngagementInvitationResponse getInvitation(String
invitationId) {

    GetEngagementInvitationRequest getRequest =
GetEngagementInvitationRequest.builder()
         .catalog(Constants.CATALOG_TO_USE)
         .identifier(invitationId)
         .build();

 GetEngagementInvitationResponse response =
client.getEngagementInvitation(getRequest);

       return response;
}

static StartEngagementByAcceptingInvitationTaskResponse getResponse(String
invitationId) {

 if ( getInvitation(invitationId).status().equals(InvitationStatus.PENDING)) {
  StartEngagementByAcceptingInvitationTaskRequest acceptOpportunityRequest =
    StartEngagementByAcceptingInvitationTaskRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
```

```
            .identifier(invitationId)
            .clientToken(clientToken)
            .build();

   StartEngagementByAcceptingInvitationTaskResponse response =
 client.startEngagementByAcceptingInvitationTask(acceptOpportunityRequest);
         return response;
  }
  return null;
 }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[StartEngagementByAcceptingInvitationTask](#)中的。

`StartEngagementFromOpportunityTask`

以下代码示例演示了如何使用 `StartEngagementFromOpportunityTask`。

适用于 Java 的 SDK 2.x

通过接受互动邀请并在合作伙伴的系统中创建相应的机会，从现有机会启动互动流程。

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import software.amazon.awssdk.services.partnercentralselling.model.AwsSubmission;
import
 software.amazon.awssdk.services.partnercentralselling.model.SalesInvolvementType;
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementFromOpportunityT
import
 software.amazon.awssdk.services.partnercentralselling.model.StartEngagementFromOpportunityT
```

```
import software.amazon.awssdk.services.partnercentralselling.model.Visibility;

/*
 * Purpose
 * PC-API-01 Partner Originated (PO) opp submission(Start Engagement From
 Opportunity Task for AO Originated Opportunity)
 */

public class StartEngagementFromOpportunityTask {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

    public static void main(String[] args) {

     String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

     StartEngagementFromOpportunityTaskResponse response =
 getResponse(opportunityId);

     ReferenceCodesUtils.formatOutput(response);
    }

 static StartEngagementFromOpportunityTaskResponse getResponse(String opportunityId)
 {

  StartEngagementFromOpportunityTaskRequest submitOpportunityRequest =
 StartEngagementFromOpportunityTaskRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .identifier(opportunityId)
          .clientToken("test-annjqwesdsd99")

 .awsSubmission(AwsSubmission.builder().involvementType(SalesInvolvementType.CO_SELL).visibi
          .build();

  StartEngagementFromOpportunityTaskResponse response =
 client.startEngagementFromOpportunityTask(submitOpportunityRequest);

        return response;
 }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartEngagementFromOpportunityTask中的。

`UpdateOpportunity`

以下代码示例演示了如何使用 `UpdateOpportunity`。

适用于 Java 的 SDK 2.x

更新机会。

```
package org.example;

import java.time.Instant;
import java.util.ArrayList;
import java.util.List;

import static org.example.utils.Constants.*;

import org.example.entity.Root;
import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;
import org.example.utils.StringSerializer;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import software.amazon.awssdk.services.partnercentralselling.model.Account;
import software.amazon.awssdk.services.partnercentralselling.model.Address;
import software.amazon.awssdk.services.partnercentralselling.model.Contact;
import software.amazon.awssdk.services.partnercentralselling.model.Customer;
import
 software.amazon.awssdk.services.partnercentralselling.model.ExpectedCustomerSpend;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.GetOpportunityResponse;
import software.amazon.awssdk.services.partnercentralselling.model.LifeCycle;
import software.amazon.awssdk.services.partnercentralselling.model.Marketing;
```

```
import software.amazon.awssdk.services.partnercentralselling.model.NextStepsHistory;
import software.amazon.awssdk.services.partnercentralselling.model.Project;
import software.amazon.awssdk.services.partnercentralselling.model.ReviewStatus;
import
 software.amazon.awssdk.services.partnercentralselling.model.UpdateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.UpdateOpportunityResponse;

import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.ToNumberPolicy;

/*
 * Purpose
 * PC-API-02/06 Update opportunity when LifeCycle.ReviewStatus is not Submitted or
 In-Review
 */

public class UpdateOpportunity {

 static final Gson GSON = new GsonBuilder()
   .setObjectToNumberStrategy(ToNumberPolicy.LAZILY_PARSED_NUMBER)
   .registerTypeAdapter(String.class, new StringSerializer())
   .create();

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();

 static String OPPORTUNITY_ORIGIN = ORIGIN_PARTNER_ORIGINATED;

 public static void main(String[] args) {

  String inputFile = "updateOpportunity.json";

  if (args.length > 0)
   inputFile = args[0];

  UpdateOpportunityResponse response = updateOpportunity(inputFile);

  client.close();
 }
```

```
public static GetOpportunityResponse getResponse(String opportunityId) {

       GetOpportunityRequest getOpportunityRequest =
GetOpportunityRequest.builder()
   .catalog(Constants.CATALOG_TO_USE)
         .identifier(opportunityId)
         .build();

       GetOpportunityResponse response =
client.getOpportunity(getOpportunityRequest);
       System.out.println(opportunityId + ":" + response);
       return response;
}

public static UpdateOpportunityResponse updateOpportunity(String inputFile) {

 String inputString = ReferenceCodesUtils.readInputFileToString(inputFile);

 Root root = GSON.fromJson(inputString, Root.class);
 GetOpportunityResponse response = getResponse(root.identifier);

 if (response != null
   && response.lifeCycle() != null
   && response.lifeCycle().reviewStatus() != null
   && response.lifeCycle().reviewStatus() != ReviewStatus.SUBMITTED
   && response.lifeCycle().reviewStatus() != ReviewStatus.IN_REVIEW) {

  List<NextStepsHistory> nextStepsHistories = new ArrayList<NextStepsHistory>();
  if ( root.lifeCycle != null && root.lifeCycle.nextStepsHistories != null) {
   for (org.example.entity.NextStepsHistory nextStepsHistoryJson :
root.lifeCycle.nextStepsHistories) {
    NextStepsHistory nextStepsHistory = NextStepsHistory.builder()
      .time(Instant.parse(nextStepsHistoryJson.time))
      .value(nextStepsHistoryJson.value)
                  .build();
    nextStepsHistories.add(nextStepsHistory);
   }
  }

  LifeCycle lifeCycle = null;
  if ( root.lifeCycle != null ) {
   lifeCycle = LifeCycle.builder()
    .closedLostReason(root.lifeCycle.closedLostReason)
```

```
    .nextSteps(root.lifeCycle.nextSteps)
    .nextStepsHistory(nextStepsHistories)
    .reviewComments(root.lifeCycle.reviewComments)
    .reviewStatus(root.lifeCycle.reviewStatus)
    .reviewStatusReason(root.lifeCycle.reviewStatusReason)
    .stage(root.lifeCycle.stage)
    .targetCloseDate(root.lifeCycle.targetCloseDate)
    .build();
  }

  Marketing marketing = null;
  if ( root.marketing != null ) {
   marketing = Marketing.builder()
     .awsFundingUsed(root.marketing.awsFundingUsed)
     .campaignName(root.marketing.campaignName)
     .channels(root.marketing.channels)
     .source(root.marketing.source)
     .useCases(root.marketing.useCases)
     .build();

  }

  Address address = null;
  if (root.customer != null && root.customer.account != null &&
root.customer.account.address != null) {
   address = Address.builder().postalCode(root.customer.account.address.postalCode)
     .stateOrRegion(root.customer.account.address.stateOrRegion)
     .countryCode(root.customer.account.address.countryCode).build();
  }

  Account account = null;
  if (root.customer != null && root.customer.account != null) {
   account = Account.builder().address(address).duns(root.customer.account.duns)

.industry(root.customer.account.industry).companyName(root.customer.account.companyName)
     .websiteUrl(root.customer.account.websiteUrl).build();
  }

  List<Contact> contacts = new ArrayList<Contact>();
  if ( root.customer != null && root.customer.contacts != null) {
   for (org.example.entity.Contact jsonContact : root.customer.contacts) {
    Contact contact = Contact.builder()
                .email(jsonContact.email)
                .firstName(jsonContact.firstName)
```

```
                    .lastName(jsonContact.lastName)
                    .phone(jsonContact.phone)
                    .businessTitle(jsonContact.businessTitle)
                    .build();
     contacts.add(contact);
    }
   }

   Customer customer =
Customer.builder().account(account).contacts(contacts).build();

   List<ExpectedCustomerSpend> expectedCustomerSpends = new
ArrayList<ExpectedCustomerSpend>();
   if ( root.project != null && root.project.expectedCustomerSpend != null) {
    for (org.example.entity.ExpectedCustomerSpend expectedCustomerSpendJson :
root.project.expectedCustomerSpend) {
     ExpectedCustomerSpend expectedCustomerSpend = null;
     expectedCustomerSpend = ExpectedCustomerSpend.builder()
       .amount(expectedCustomerSpendJson.amount)
       .currencyCode(expectedCustomerSpendJson.currencyCode)
       .frequency(expectedCustomerSpendJson.frequency)
       .targetCompany(expectedCustomerSpendJson.targetCompany)
       .build();
     expectedCustomerSpends.add(expectedCustomerSpend);
    }
         }

   Project project = null;
   if (root.project != null) {
    project = Project.builder().title(root.project.title)
      .customerBusinessProblem(root.project.customerBusinessProblem)

.customerUseCase(root.project.customerUseCase).deliveryModels(root.project.deliveryModels)
      .expectedCustomerSpend(expectedCustomerSpends)

.salesActivities(root.project.salesActivities).competitorName(root.project.competitorName)
      .otherSolutionDescription(root.project.otherSolutionDescription).build();
   }

   // Building the Actual CreateOpportunity Request
   UpdateOpportunityRequest updateOpportunityRequest =
UpdateOpportunityRequest.builder().catalog(root.catalog)

.identifier(root.identifier).lastModifiedDate(Instant.parse(root.lastModifiedDate))
```

```
 .primaryNeedsFromAwsWithStrings(root.primaryNeedsFromAws).opportunityType(root.opportunityT
     .lifeCycle(lifeCycle)
     .customer(customer)
     .project(project)
     .partnerOpportunityIdentifier(root.partnerOpportunityIdentifier)
     .marketing(marketing)
     .nationalSecurity(root.nationalSecurity)
     .opportunityType(root.opportunityType)
     .build();

   UpdateOpportunityResponse updateResponse =
 client.updateOpportunity(updateOpportunityRequest);
   System.out.println("Successfully updated opportunity: " + updateResponse);

   return updateResponse;
  } else {
   System.out.println("Opportunity cannot be updated.");
   return null;
  }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考UpdateOpportunity中的。

场景

更新机会的关联实体

以下代码示例展示了如何：

- 取消与旧实体的关联。
- 关联新实体。

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Scenarios 存储库中查找完整示例，了解如何进行设置和运行。

更新机会的关联实体

```
package org.example;

import static org.example.utils.Constants.*;

import org.example.utils.Constants;
import org.example.utils.ReferenceCodesUtils;

import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.partnercentralselling.PartnerCentralSellingClient;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.AssociateOpportunityResponse;
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityRequest;
import
 software.amazon.awssdk.services.partnercentralselling.model.DisassociateOpportunityResponse

/*
Purpose
PC-API -17 Replacing a solution
*/

public class ReplaceSolution {

 static PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(DefaultCredentialsProvider.create())
            .httpClient(ApacheHttpClient.builder().build())
            .build();
```

```
   public static void main(String[] args) {

    String opportunityId = args.length > 0 ? args[0] : OPPORTUNITY_ID;

    String entityType = "Solutions";
    String originalEntityIdentifier = "S-0000000";
    String newEntityIdentifier = "S-0011111";

    disassociateOppornititityResponse(opportunityId, entityType,
originalEntityIdentifier );
    AssociateOpportunityResponse associateOpportunityResponse =
associateOpportunityResponse(opportunityId, entityType, newEntityIdentifier );

    ReferenceCodesUtils.formatOutput(associateOpportunityResponse);
   }

private static AssociateOpportunityResponse associateOpportunityResponse(String
opportunityId, String entityType, String entityIdentifier) {

       AssociateOpportunityRequest associateOpportunityRequest =
AssociateOpportunityRequest.builder()
   .catalog(Constants.CATALOG_TO_USE)
         .opportunityIdentifier(opportunityId)
         .relatedEntityType(entityType)
         .relatedEntityIdentifier(entityIdentifier)
         .build();

       AssociateOpportunityResponse response =
client.associateOpportunity(associateOpportunityRequest);

       return response;
}

private static DisassociateOpportunityResponse
disassociateOppornititityResponse(String opportunityId, String entityType, String
entityIdentifier) {
 PartnerCentralSellingClient client = PartnerCentralSellingClient.builder()
           .region(Region.US_EAST_1)
           .credentialsProvider(DefaultCredentialsProvider.create())
           .httpClient(ApacheHttpClient.builder().build())
           .build();
```

```
        DisassociateOpportunityRequest disassociateOpportunityRequest =
 DisassociateOpportunityRequest.builder()
    .catalog(Constants.CATALOG_TO_USE)
          .opportunityIdentifier(opportunityId)
          .relatedEntityType(entityType)
          .relatedEntityIdentifier(entityIdentifier)
          .build();

        DisassociateOpportunityResponse response =
 client.disassociateOpportunity(disassociateOpportunityRequest);

        return response;
 }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - AssociateOpportunity
  - DisassociateOpportunity

使用 SDK for Java 2.x 的 Amazon Personalize 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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从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`

以下代码示例演示了如何使用 `CreateBatchInferenceJob`。

适用于 Java 的 SDK 2.x

> **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。

```
        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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateBatchInferenceJob中的。

`CreateCampaign`

以下代码示例演示了如何使用 `CreateCampaign`。

适用于 Java 的 SDK 2.x

> **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。

```
   public static String createPersonalCompaign(PersonalizeClient personalizeClient,
String solutionVersionArn,
           String name) {

       try {
           CreateCampaignRequest createCampaignRequest =
CreateCampaignRequest.builder()
                   .minProvisionedTPS(1)
                   .solutionVersionArn(solutionVersionArn)
                   .name(name)
                   .build();

           CreateCampaignResponse campaignResponse =
personalizeClient.createCampaign(createCampaignRequest);
           System.out.println("The campaign ARN is " +
campaignResponse.campaignArn());
           return campaignResponse.campaignArn();
       } catch (PersonalizeException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return null;
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateCampaign中的。

`CreateDataset`

以下代码示例演示了如何使用 `CreateDataset`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateDataset中的。

`CreateDatasetExportJob`

以下代码示例演示了如何使用 `CreateDatasetExportJob`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateDatasetExportJob中的。

`CreateDatasetGroup`

以下代码示例演示了如何使用 `CreateDatasetGroup`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateDatasetGroup中的。

`CreateDatasetImportJob`

以下代码示例演示了如何使用 `CreateDatasetImportJob`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateDatasetImportJob中的。

`CreateEventTracker`

以下代码示例演示了如何使用 `CreateEventTracker`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateEventTracker中的。

`CreateFilter`

以下代码示例演示了如何使用 `CreateFilter`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateFilter中的。

`CreateRecommender`

以下代码示例演示了如何使用 `CreateRecommender`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateRecommender中的。

`CreateSchema`

以下代码示例演示了如何使用 `CreateSchema`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 "";
    }
```

创建含域的架构。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateSchema中的。

`CreateSolution`

以下代码示例演示了如何使用 `CreateSolution`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateSolution中的。

`CreateSolutionVersion`

以下代码示例演示了如何使用 `CreateSolutionVersion`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateSolutionVersion中的。

`DeleteCampaign`

以下代码示例演示了如何使用 `DeleteCampaign`。

适用于 Java 的 SDK 2.x

> **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。

```
    public static void deleteSpecificCampaign(PersonalizeClient personalizeClient,
 String campaignArn) {
        try {
            DeleteCampaignRequest campaignRequest = DeleteCampaignRequest.builder()
                    .campaignArn(campaignArn)
                    .build();

            personalizeClient.deleteCampaign(campaignRequest);
            System.out.println("Delete request sent successfully.");
        } catch (PersonalizeException e) {
            System.err.println("Error deleting campaign: " +
 e.awsErrorDetails().errorMessage());
            throw new RuntimeException(e);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteCampaign中的。

`DeleteEventTracker`

以下代码示例演示了如何使用 `DeleteEventTracker`。

适用于 Java 的 SDK 2.x

> **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。

```
    public static void deleteEventTracker(PersonalizeClient personalizeClient,
 String eventTrackerArn) {
        try {
            DeleteEventTrackerRequest deleteEventTrackerRequest =
 DeleteEventTrackerRequest.builder()
                    .eventTrackerArn(eventTrackerArn)
                    .build();
```

```
            int status =
personalizeClient.deleteEventTracker(deleteEventTrackerRequest).sdkHttpResponse().statusCod

            System.out.println("Status code:" + status);

        } catch (PersonalizeException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteEventTracker中的。

`DeleteSolution`

以下代码示例演示了如何使用 `DeleteSolution`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteSolution中的。

`DescribeCampaign`

以下代码示例演示了如何使用 `DescribeCampaign`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeCampaign中的。

DescribeRecipe

以下代码示例演示了如何使用 DescribeRecipe。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeRecipe中的。

DescribeSolution

以下代码示例演示了如何使用 DescribeSolution。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeSolution中的。

`ListCampaigns`

以下代码示例演示了如何使用 `ListCampaigns`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListCampaigns中的。

`ListDatasetGroups`

以下代码示例演示了如何使用 `ListDatasetGroups`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListDatasetGroups中的。

`ListRecipes`

以下代码示例演示了如何使用 `ListRecipes`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListRecipes中的。

`ListSolutions`

以下代码示例演示了如何使用 `ListSolutions`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListSolutions中的。

UpdateCampaign

以下代码示例演示了如何使用 UpdateCampaign。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考UpdateCampaign中的。

使用 SDK for Java 2.x 的 Amazon Personalize Events 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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从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`

以下代码示例演示了如何使用 `PutEvents`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutEvents中的。

PutUsers

以下代码示例演示了如何使用 PutUsers。

适用于 Java 的 SDK 2.x

> **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。

```
        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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutUsers中的。

使用 SDK for Java 2.x 的 Amazon Personalize Runtime 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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从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`

以下代码示例演示了如何使用 `GetPersonalizedRanking`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[GetPersonalizedRanking]中的。

`GetRecommendations`

以下代码示例演示了如何使用 `GetRecommendations`。

适用于 Java 的 SDK 2.x

> **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。

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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[GetRecommendations](#)中的。

使用 SDK for 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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从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`

以下代码示例演示了如何使用 `CreateApp`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateApp中的。

`CreateCampaign`

以下代码示例演示了如何使用 `CreateCampaign`。

适用于 Java 的 SDK 2.x

> **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。

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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateCampaign中的。

CreateExportJob

以下代码示例演示了如何使用 CreateExportJob。

适用于 Java 的 SDK 2.x

> **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。

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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateExportJob中的。

`CreateImportJob`

以下代码示例演示了如何使用 `CreateImportJob`。

适用于 Java 的 SDK 2.x

> **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。

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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateImportJob中的。

`CreateSegment`

以下代码示例演示了如何使用 `CreateSegment`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateSegment中的。

`DeleteApp`

以下代码示例演示了如何使用 `DeleteApp`。

适用于 Java 的 SDK 2.x

> **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。

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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteApp中的。

`DeleteEndpoint`

以下代码示例演示了如何使用 `DeleteEndpoint`。

适用于 Java 的 SDK 2.x

> **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。

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();

        deletePinEncpoint(pinpoint, appId, endpointId);
        pinpoint.close();
    }

    public static void deletePinEncpoint(PinpointClient pinpoint, String appId,
 String endpointId) {
        try {
            DeleteEndpointRequest appRequest = DeleteEndpointRequest.builder()
                    .applicationId(appId)
                    .endpointId(endpointId)
                    .build();

            DeleteEndpointResponse result = pinpoint.deleteEndpoint(appRequest);
            String id = result.endpointResponse().id();
            System.out.println("The deleted endpoint id  " + id);

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteEndpoint中的。

GetEndpoint

以下代码示例演示了如何使用 GetEndpoint。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[GetEndpoint](#)中的。

GetSegments

以下代码示例演示了如何使用 GetSegments。

适用于 Java 的 SDK 2.x

> **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。

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("Segement " + segment.id() + " " + segment.name() +
 " " + segment.lastModifiedDate());
            }

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetSegments中的。

GetSmsChannel

以下代码示例演示了如何使用 GetSmsChannel。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetSmsChannel中的。

`GetUserEndpoints`

以下代码示例演示了如何使用 `GetUserEndpoints`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetUserEndpoints中的。

SendMessages

以下代码示例演示了如何使用 SendMessages。

适用于 Java 的 SDK 2.x

> **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。

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();
        }
    }
}
```

发送短信。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
import software.amazon.awssdk.services.pinpoint.model.SMSMessage;
```

```
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesResponse;
import software.amazon.awssdk.services.pinpoint.model.MessageResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;
import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessage {

        // The type of SMS message that you want to send. If you plan to send
        // time-sensitive content, specify TRANSACTIONAL. If you plan to send
        // marketing-related content, specify PROMOTIONAL.
        public static String messageType = "TRANSACTIONAL";

        // The registered keyword associated with the originating short code.
        public static String registeredKeyword = "myKeyword";

        // The sender ID to use when sending the message. Support for sender ID
        // varies by country or region. For more information, see
        // https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-
countries.html
        public static String senderId = "MySenderID";

        public static void main(String[] args) {
                final String usage = """

                                Usage:   <message> <appId> <originationNumber>
 <destinationNumber>\s

                                Where:
                                  message - The body of the message to send.
                                  appId - The Amazon Pinpoint project/application ID
 to use when you send this message.
```

```
                                originationNumber - The phone number or short code
that you specify has to be associated with your Amazon Pinpoint account. For best
results, specify long codes in E.164 format (for example, +1-555-555-5654).
                                destinationNumber - The recipient's phone number.
For best results, you should specify the phone number in E.164 format (for example,
+1-555-555-5654).\s
                                """;

                if (args.length != 4) {
                        System.out.println(usage);
                        System.exit(1);
                }

                String message = args[0];
                String appId = args[1];
                String originationNumber = args[2];
                String destinationNumber = args[3];
                System.out.println("Sending a message");
                PinpointClient pinpoint = PinpointClient.builder()
                                .region(Region.US_EAST_1)
                                .build();

                sendSMSMessage(pinpoint, message, appId, originationNumber,
destinationNumber);
                pinpoint.close();
        }

        public static void sendSMSMessage(PinpointClient pinpoint, String message,
String appId,
                        String originationNumber,
                        String destinationNumber) {
                try {
                        Map<String, AddressConfiguration> addressMap = new
HashMap<String, AddressConfiguration>();
                        AddressConfiguration addConfig =
AddressConfiguration.builder()
                                        .channelType(ChannelType.SMS)
                                        .build();

                        addressMap.put(destinationNumber, addConfig);
                        SMSMessage smsMessage = SMSMessage.builder()
                                        .body(message)
                                        .messageType(messageType)
                                        .originationNumber(originationNumber)
```

```
                                        .senderId(senderId)
                                        .keyword(registeredKeyword)
                                        .build();

                        // Create a DirectMessageConfiguration object.
                        DirectMessageConfiguration direct =
 DirectMessageConfiguration.builder()
                                        .smsMessage(smsMessage)
                                        .build();

                        MessageRequest msgReq = MessageRequest.builder()
                                        .addresses(addressMap)
                                        .messageConfiguration(direct)
                                        .build();

                        // create a SendMessagesRequest object
                        SendMessagesRequest request = SendMessagesRequest.builder()
                                        .applicationId(appId)
                                        .messageRequest(msgReq)
                                        .build();

                        SendMessagesResponse response =
 pinpoint.sendMessages(request);
                        MessageResponse msg1 = response.messageResponse();
                        Map map1 = msg1.result();

                        // Write out the result of sendMessage.
                        map1.forEach((k, v) -> System.out.println((k + ":" + v)));

                } catch (PinpointException e) {
                        System.err.println(e.awsErrorDetails().errorMessage());
                        System.exit(1);
                }
        }
}
```

发送批处理短信。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpoint.PinpointClient;
import software.amazon.awssdk.services.pinpoint.model.DirectMessageConfiguration;
```

```
import software.amazon.awssdk.services.pinpoint.model.SMSMessage;
import software.amazon.awssdk.services.pinpoint.model.AddressConfiguration;
import software.amazon.awssdk.services.pinpoint.model.ChannelType;
import software.amazon.awssdk.services.pinpoint.model.MessageRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesRequest;
import software.amazon.awssdk.services.pinpoint.model.SendMessagesResponse;
import software.amazon.awssdk.services.pinpoint.model.MessageResponse;
import software.amazon.awssdk.services.pinpoint.model.PinpointException;

import java.util.HashMap;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendMessageBatch {

    // The type of SMS message that you want to send. If you plan to send
    // time-sensitive content, specify TRANSACTIONAL. If you plan to send
    // marketing-related content, specify PROMOTIONAL.
    public static String messageType = "TRANSACTIONAL";

    // The registered keyword associated with the originating short code.
    public static String registeredKeyword = "myKeyword";

    // The sender ID to use when sending the message. Support for sender ID
    // varies by country or region. For more information, see
    // https://docs.aws.amazon.com/pinpoint/latest/userguide/channels-sms-
countries.html
    public static String senderId = "MySenderID";

    public static void main(String[] args) {
        final String usage = """

                Usage:   <message> <appId> <originationNumber> <destinationNumber>
 <destinationNumber1>\s

                Where:
                  message - The body of the message to send.
```

```
                appId - The Amazon Pinpoint project/application ID to use when you
send this message.
                originationNumber - The phone number or short code that you
specify has to be associated with your Amazon Pinpoint account. For best results,
specify long codes in E.164 format (for example, +1-555-555-5654).
                destinationNumber - The recipient's phone number.  For best
results, you should specify the phone number in E.164 format (for example,
+1-555-555-5654).
                destinationNumber1 - The second recipient's phone number.  For
best results, you should specify the phone number in E.164 format (for example,
+1-555-555-5654).\s
               """;

       if (args.length != 5) {
           System.out.println(usage);
           System.exit(1);
       }

       String message = args[0];
       String appId = args[1];
       String originationNumber = args[2];
       String destinationNumber = args[3];
       String destinationNumber1 = args[4];
       System.out.println("Sending a message");
       PinpointClient pinpoint = PinpointClient.builder()
               .region(Region.US_EAST_1)
               .build();

       sendSMSMessage(pinpoint, message, appId, originationNumber,
destinationNumber, destinationNumber1);
       pinpoint.close();
   }

   public static void sendSMSMessage(PinpointClient pinpoint, String message,
String appId,
                                     String originationNumber,
                                     String destinationNumber, String
destinationNumber1) {
       try {
           Map<String, AddressConfiguration> addressMap = new HashMap<String,
AddressConfiguration>();
           AddressConfiguration addConfig = AddressConfiguration.builder()
                   .channelType(ChannelType.SMS)
                   .build();
```

```
            // Add an entry to the Map object for each number to whom you want to
 send a
            // message.
            addressMap.put(destinationNumber, addConfig);
            addressMap.put(destinationNumber1, addConfig);
            SMSMessage smsMessage = SMSMessage.builder()
                    .body(message)
                    .messageType(messageType)
                    .originationNumber(originationNumber)
                    .senderId(senderId)
                    .keyword(registeredKeyword)
                    .build();

            // Create a DirectMessageConfiguration object.
            DirectMessageConfiguration direct = DirectMessageConfiguration.builder()
                    .smsMessage(smsMessage)
                    .build();

            MessageRequest msgReq = MessageRequest.builder()
                    .addresses(addressMap)
                    .messageConfiguration(direct)
                    .build();

            // Create a SendMessagesRequest object.
            SendMessagesRequest request = SendMessagesRequest.builder()
                    .applicationId(appId)
                    .messageRequest(msgReq)
                    .build();

            SendMessagesResponse response = pinpoint.sendMessages(request);
            MessageResponse msg1 = response.messageResponse();
            Map map1 = msg1.result();

            // Write out the result of sendMessage.
            map1.forEach((k, v) -> System.out.println((k + ":" + v)));

        } catch (PinpointException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SendMessages中的。

`UpdateEndpoint`

以下代码示例演示了如何使用 `UpdateEndpoint`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考UpdateEndpoint中的。

使用 SDK for Java 2.x 的 Amazon Pinpoint 短信和语音 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 短信和语音 API 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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从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`

以下代码示例演示了如何使用 `SendVoiceMessage`。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.pinpointsmsvoice.PinpointSmsVoiceClient;
```

```
import software.amazon.awssdk.services.pinpointsmsvoice.model.SSMLMessageType;
import software.amazon.awssdk.services.pinpointsmsvoice.model.VoiceMessageContent;
import
 software.amazon.awssdk.services.pinpointsmsvoice.model.SendVoiceMessageRequest;
import
 software.amazon.awssdk.services.pinpointsmsvoice.model.PinpointSmsVoiceException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SendVoiceMessage {

    // The Amazon Polly voice that you want to use to send the message. For a list
    // of voices, see https://docs.aws.amazon.com/polly/latest/dg/voicelist.html
    static final String voiceName = "Matthew";

    // The language to use when sending the message. For a list of supported
    // languages, see
    // https://docs.aws.amazon.com/polly/latest/dg/SupportedLanguage.html
    static final String languageCode = "en-US";

    // The content of the message. This example uses SSML to customize and control
    // certain aspects of the message, such as by adding pauses and changing
    // phonation. The message can't contain any line breaks.
    static final String ssmlMessage = "<speak>This is a test message sent from "
            + "<emphasis>Amazon Pinpoint</emphasis> "
            + "using the <break strength='weak'/>AWS "
            + "SDK for Java. "
            + "<amazon:effect phonation='soft'>Thank "
            + "you for listening.</amazon:effect></speak>";

    public static void main(String[] args) {

        final String usage = """
```

```
                Usage:   <originationNumber> <destinationNumber>\s

                Where:
                  originationNumber - The phone number or short code that you
specify has to be associated with your Amazon Pinpoint account. For best results,
specify long codes in E.164 format (for example, +1-555-555-5654).
                  destinationNumber - The recipient's phone number.  For best
results, you should specify the phone number in E.164 format (for example,
+1-555-555-5654).\s
                """;

       if (args.length != 2) {
           System.out.println(usage);
           System.exit(1);
       }
       String originationNumber = args[0];
       String destinationNumber = args[1];
       System.out.println("Sending a voice message");

       // Set the content type to application/json.
       List<String> listVal = new ArrayList<>();
       listVal.add("application/json");
       Map<String, List<String>> values = new HashMap<>();
       values.put("Content-Type", listVal);

       ClientOverrideConfiguration config2 = ClientOverrideConfiguration.builder()
               .headers(values)
               .build();

       PinpointSmsVoiceClient client = PinpointSmsVoiceClient.builder()
               .overrideConfiguration(config2)
               .region(Region.US_EAST_1)
               .build();

       sendVoiceMsg(client, originationNumber, destinationNumber);
       client.close();
   }

   public static void sendVoiceMsg(PinpointSmsVoiceClient client, String
originationNumber,
                                   String destinationNumber) {
       try {
           SSMLMessageType ssmlMessageType = SSMLMessageType.builder()
                   .languageCode(languageCode)
```

```
                    .text(ssmlMessage)
                    .voiceId(voiceName)
                    .build();

            VoiceMessageContent content = VoiceMessageContent.builder()
                    .ssmlMessage(ssmlMessageType)
                    .build();

            SendVoiceMessageRequest voiceMessageRequest =
 SendVoiceMessageRequest.builder()
                    .destinationPhoneNumber(destinationNumber)
                    .originationPhoneNumber(originationNumber)
                    .content(content)
                    .build();

            client.sendVoiceMessage(voiceMessageRequest);
            System.out.println("The message was sent successfully.");

        } catch (PinpointSmsVoiceException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SendVoiceMessage中的。

使用 SDK for 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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从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

`DescribeVoices`

以下代码示例演示了如何使用 `DescribeVoices`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeVoices中的。

`ListLexicons`

以下代码示例演示了如何使用 `ListLexicons`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListLexicons中的。

SynthesizeSpeech

以下代码示例演示了如何使用 SynthesizeSpeech。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SynthesizeSpeech中的。

场景

创建用于分析客户反馈的应用程序

以下代码示例说明如何创建应用程序来分析客户意见卡、翻译其母语、确定其情绪并根据译后的文本生成音频文件。

适用于 Java 的 SDK 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 SDK for Java 2.x 的 Amazon RDS 示例

以下代码示例向您展示了如何在 Amazon RDS 中使用来执行操作和实现常见场景。 AWS SDK for Java 2.x

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 基本功能
- 操作
- 场景

- 无服务器示例

开始使用

开始使用 Amazon RDS

以下代码示例展示了如何开始使用 Amazon RDS。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DBInstances中的描述。

基本功能

了解基本功能

以下代码示例展示了如何：

- 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。

适用于 Java 的 SDK 2.x

> **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。

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)
            .build();
    }

    public static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
```

```
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
                .secretId(secretName)
                .build();

        GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }

    // Delete the parameter group after database has been deleted.
    // An exception is thrown if you attempt to delete the para group while database
    // exists.
    public static void deleteParaGroup(RdsClient rdsClient, String dbGroupName,
String dbARN)
            throws InterruptedException {
        try {
            boolean isDataDel = false;
            boolean didFind;
            String instanceARN;

            // Make sure that the database has been deleted.
            while (!isDataDel) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances();
                List<DBInstance> instanceList = response.dbInstances();
                int listSize = instanceList.size();
                didFind = false;
                int index = 1;
                for (DBInstance instance : instanceList) {
                    instanceARN = instance.dbInstanceArn();
                    if (instanceARN.compareTo(dbARN) == 0) {
                        System.out.println(dbARN + " still exists");
                        didFind = true;
                    }
                    if ((index == listSize) && (!didFind)) {
                        // Went through the entire list and did not find the
database ARN.
                        isDataDel = true;
                    }
                    Thread.sleep(sleepTime * 1000);
                    index++;
                }
            }
```

```
            // Delete the para group.
            DeleteDbParameterGroupRequest parameterGroupRequest =
DeleteDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .build();

            rdsClient.deleteDBParameterGroup(parameterGroupRequest);
            System.out.println(dbGroupName + " was deleted.");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Delete the DB instance.
    public static void deleteDatabaseInstance(RdsClient rdsClient, String
dbInstanceIdentifier) {
        try {
            DeleteDbInstanceRequest deleteDbInstanceRequest =
DeleteDbInstanceRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .deleteAutomatedBackups(true)
                    .skipFinalSnapshot(true)
                    .build();

            DeleteDbInstanceResponse response =
rdsClient.deleteDBInstance(deleteDbInstanceRequest);
            System.out.print("The status of the database is " +
response.dbInstance().dbInstanceStatus());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Waits until the snapshot instance is available.
    public static void waitForSnapshotReady(RdsClient rdsClient, String
dbInstanceIdentifier,
            String dbSnapshotIdentifier) {
        try {
            boolean snapshotReady = false;
            String snapshotReadyStr;
```

```
            System.out.println("Waiting for the snapshot to become available.");

            DescribeDbSnapshotsRequest snapshotsRequest =
DescribeDbSnapshotsRequest.builder()
                    .dbSnapshotIdentifier(dbSnapshotIdentifier)
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            while (!snapshotReady) {
                DescribeDbSnapshotsResponse response =
rdsClient.describeDBSnapshots(snapshotsRequest);
                List<DBSnapshot> snapshotList = response.dbSnapshots();
                for (DBSnapshot snapshot : snapshotList) {
                    snapshotReadyStr = snapshot.status();
                    if (snapshotReadyStr.contains("available")) {
                        snapshotReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }

            System.out.println("The Snapshot is available!");
        } catch (RdsException | InterruptedException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Create an Amazon RDS snapshot.
    public static void createSnapshot(RdsClient rdsClient, String
dbInstanceIdentifier, String dbSnapshotIdentifier) {
        try {
            CreateDbSnapshotRequest snapshotRequest =
CreateDbSnapshotRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .dbSnapshotIdentifier(dbSnapshotIdentifier)
                    .build();

            CreateDbSnapshotResponse response =
rdsClient.createDBSnapshot(snapshotRequest);
            System.out.println("The Snapshot id is " +
response.dbSnapshot().dbiResourceId());
```

```
        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            String endpoint = "";
            while (!instanceReady) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
                List<DBInstance> instanceList = response.dbInstances();
                for (DBInstance instance : instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus();
                    if (instanceReadyStr.contains("available")) {
                        endpoint = instance.endpoint().address();
                        instanceReady = true;
                    } else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database instance is available! The connection
endpoint is " + endpoint);

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

```
   // Create a database instance and return the ARN of the database.
   public static String createDatabaseInstance(RdsClient rdsClient,
           String dbGroupName,
           String dbInstanceIdentifier,
           String dbName,
           String userName,
           String userPassword) {

       try {
           CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
               .dbInstanceIdentifier(dbInstanceIdentifier)
               .allocatedStorage(100)
               .dbName(dbName)
               .engine("mysql")
               .dbInstanceClass("db.t3.medium") // Updated to a supported class
               .engineVersion("8.0.32")         // Updated to a supported version
               .storageType("gp2")             // Changed to General Purpose SSD
(gp2)
               .masterUsername(userName)
               .masterUserPassword(userPassword)
               .build();

           CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
           System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());
           return response.dbInstance().dbInstanceArn();

       } catch (RdsException e) {
           System.out.println(e.getLocalizedMessage());
           System.exit(1);
       }

       return "";
   }

   // Get a list of micro instances.
   public static void getMicroInstances(RdsClient rdsClient) {
       try {
           DescribeOrderableDbInstanceOptionsRequest dbInstanceOptionsRequest =
DescribeOrderableDbInstanceOptionsRequest
                   .builder()
                   .engine("mysql")
```

```
                    .build();

            DescribeOrderableDbInstanceOptionsResponse response = rdsClient
                    .describeOrderableDBInstanceOptions(dbInstanceOptionsRequest);
            List<OrderableDBInstanceOption> orderableDBInstances =
response.orderableDBInstanceOptions();
            for (OrderableDBInstanceOption dbInstanceOption : orderableDBInstances)
{
                System.out.println("The engine version is " +
dbInstanceOption.engineVersion());
                System.out.println("The engine description is " +
dbInstanceOption.engine());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Get a list of allowed engine versions.
    public static void getAllowedEngines(RdsClient rdsClient, String
dbParameterGroupFamily) {
        try {
            DescribeDbEngineVersionsRequest versionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .engine("mysql")
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(versionsRequest);
            List<DBEngineVersion> dbEngines = response.dbEngineVersions();
            for (DBEngineVersion dbEngine : dbEngines) {
                System.out.println("The engine version is " +
dbEngine.engineVersion());
                System.out.println("The engine description is " +
dbEngine.dbEngineDescription());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
```

```
    }

    // Modify auto_increment_offset and auto_increment_increment parameters.
    public static void modifyDBParas(RdsClient rdsClient, String dbGroupName) {
        try {
            Parameter parameter1 = Parameter.builder()
                    .parameterName("auto_increment_offset")
                    .applyMethod("immediate")
                    .parameterValue("5")
                    .build();

            List<Parameter> paraList = new ArrayList<>();
            paraList.add(parameter1);
            ModifyDbParameterGroupRequest groupRequest =
ModifyDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .parameters(paraList)
                    .build();

            ModifyDbParameterGroupResponse response =
rdsClient.modifyDBParameterGroup(groupRequest);
            System.out.println("The parameter group " +
response.dbParameterGroupName() + " was successfully modified");

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Retrieve parameters in the group.
    public static void describeDbParameters(RdsClient rdsClient, String dbGroupName,
int flag) {
        try {
            DescribeDbParametersRequest dbParameterGroupsRequest;
            if (flag == 0) {
                dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                        .dbParameterGroupName(dbGroupName)
                        .build();
            } else {
                dbParameterGroupsRequest = DescribeDbParametersRequest.builder()
                        .dbParameterGroupName(dbGroupName)
                        .source("user")
                        .build();
```

```
            }

            DescribeDbParametersResponse response =
rdsClient.describeDBParameters(dbParameterGroupsRequest);
            List<Parameter> dbParameters = response.parameters();
            String paraName;
            for (Parameter para : dbParameters) {
                // Only print out information about either auto_increment_offset or
                // auto_increment_increment.
                paraName = para.parameterName();
                if ((paraName.compareTo("auto_increment_offset") == 0)
                        || (paraName.compareTo("auto_increment_increment ") == 0)) {
                    System.out.println("*** The parameter name is  " + paraName);
                    System.out.println("*** The parameter value is  " +
para.parameterValue());
                    System.out.println("*** The parameter data type is " +
para.dataType());
                    System.out.println("*** The parameter description is " +
para.description());
                    System.out.println("*** The parameter allowed values  is " +
para.allowedValues());
                }
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDbParameterGroups(RdsClient rdsClient, String
dbGroupName) {
        try {
            DescribeDbParameterGroupsRequest groupsRequest =
DescribeDbParameterGroupsRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .maxRecords(20)
                    .build();

            DescribeDbParameterGroupsResponse response =
rdsClient.describeDBParameterGroups(groupsRequest);
            List<DBParameterGroup> groups = response.dbParameterGroups();
            for (DBParameterGroup group : groups) {
```

```
                System.out.println("The group name is " +
group.dbParameterGroupName());
                System.out.println("The group description is " +
group.description());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void createDBParameterGroup(RdsClient rdsClient, String
dbGroupName, String dbParameterGroupFamily) {
        try {
            CreateDbParameterGroupRequest groupRequest =
CreateDbParameterGroupRequest.builder()
                    .dbParameterGroupName(dbGroupName)
                    .dbParameterGroupFamily(dbParameterGroupFamily)
                    .description("Created by using the AWS SDK for Java")
                    .build();

            CreateDbParameterGroupResponse response =
rdsClient.createDBParameterGroup(groupRequest);
            System.out.println("The group name is " +
response.dbParameterGroup().dbParameterGroupName());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeDBEngines(RdsClient rdsClient) {
        try {
            DescribeDbEngineVersionsRequest engineVersionsRequest =
DescribeDbEngineVersionsRequest.builder()
                    .defaultOnly(true)
                    .engine("mysql")
                    .maxRecords(20)
                    .build();

            DescribeDbEngineVersionsResponse response =
rdsClient.describeDBEngineVersions(engineVersionsRequest);
```

```
            List<DBEngineVersion> engines = response.dbEngineVersions();

            // Get all DBEngineVersion objects.
            for (DBEngineVersion engineOb : engines) {
                System.out.println("The name of the DB parameter group family for
 the database engine is "
                        + engineOb.dbParameterGroupFamily());
                System.out.println("The name of the database engine " +
 engineOb.engine());
                System.out.println("The version number of the database engine " +
 engineOb.engineVersion());
            }

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - 创建DBInstance
  - 创建DBParameter群组
  - 创建DBSnapshot
  - 删除DBInstance
  - 删除DBParameter群组
  - 描述DBEngine版本
  - 描述DBInstances
  - 描述DBParameter群组
  - 描述DBParameters
  - 描述DBSnapshots
  - DescribeOrderableDBInstanceOptions
  - 修改DBParameter群组

操作

`CreateDBInstance`

以下代码示例演示了如何使用 `CreateDBInstance`。

适用于 Java 的 SDK 2.x

> **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。

```
import com.google.gson.Gson;
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rds.RdsClient;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceRequest;
import software.amazon.awssdk.services.rds.model.CreateDbInstanceResponse;
import software.amazon.awssdk.services.rds.model.RdsException;
import software.amazon.awssdk.services.rds.model.DescribeDbInstancesResponse;
import software.amazon.awssdk.services.rds.model.DBInstance;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example requires an AWS Secrets Manager secret that contains the
 * database credentials. If you do not create a
 * secret, this example will not work. For more details, see:
```

```
 *
 * https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
 *
 *
 */

public class CreateDBInstance {
    public static long sleepTime = 20;

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbInstanceIdentifier> <dbName> <secretName>

                Where:
                    dbInstanceIdentifier - The database instance identifier.\s
                    dbName - The database name.\s
                    secretName - The name of the AWS Secrets Manager secret that
 contains the database credentials."
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbInstanceIdentifier = args[0];
        String dbName = args[1];
        String secretName = args[2];
        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
 User.class);
        Region region = Region.US_WEST_2;
        RdsClient rdsClient = RdsClient.builder()
                .region(region)
                .build();

        createDatabaseInstance(rdsClient, dbInstanceIdentifier, dbName,
 user.getUsername(), user.getPassword());
        waitForInstanceReady(rdsClient, dbInstanceIdentifier);
        rdsClient.close();
    }
```

```
   private static SecretsManagerClient getSecretClient() {
       Region region = Region.US_WEST_2;
       return SecretsManagerClient.builder()
               .region(region)

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

   private static String getSecretValues(String secretName) {
       SecretsManagerClient secretClient = getSecretClient();
       GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
               .secretId(secretName)
               .build();

       GetSecretValueResponse valueResponse =
secretClient.getSecretValue(valueRequest);
       return valueResponse.secretString();
   }

   public static void createDatabaseInstance(RdsClient rdsClient,
           String dbInstanceIdentifier,
           String dbName,
           String userName,
           String userPassword) {

       try {
           CreateDbInstanceRequest instanceRequest =
CreateDbInstanceRequest.builder()
               .dbInstanceIdentifier(dbInstanceIdentifier)
               .allocatedStorage(100)
               .dbName(dbName)
               .engine("mysql")
               .dbInstanceClass("db.t3.medium") // Updated to a supported class
               .engineVersion("8.0.32")         // Updated to a supported version
               .storageType("gp2")             // Changed to General Purpose SSD
(gp2)
               .masterUsername(userName)
               .masterUserPassword(userPassword)
               .build();

           CreateDbInstanceResponse response =
rdsClient.createDBInstance(instanceRequest);
```

```
            System.out.print("The status is " +
response.dbInstance().dbInstanceStatus());

        } catch (RdsException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    // Waits until the database instance is available.
    public static void waitForInstanceReady(RdsClient rdsClient, String
dbInstanceIdentifier) {
        boolean instanceReady = false;
        String instanceReadyStr;
        System.out.println("Waiting for instance to become available.");
        try {
            DescribeDbInstancesRequest instanceRequest =
DescribeDbInstancesRequest.builder()
                    .dbInstanceIdentifier(dbInstanceIdentifier)
                    .build();

            // Loop until the cluster is ready.
            while (!instanceReady) {
                DescribeDbInstancesResponse response =
rdsClient.describeDBInstances(instanceRequest);
                List<DBInstance> instanceList = response.dbInstances();
                for (DBInstance instance : instanceList) {
                    instanceReadyStr = instance.dbInstanceStatus();
                    if (instanceReadyStr.contains("available"))
                        instanceReady = true;
                    else {
                        System.out.print(".");
                        Thread.sleep(sleepTime * 1000);
                    }
                }
            }
            System.out.println("Database instance is available!");

        } catch (RdsException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

- 有关 API 的详细信息，请参阅DBInstance在 AWS SDK for Java 2.x API 参考中创建。

`CreateDBParameterGroup`

以下代码示例演示了如何使用 `CreateDBParameterGroup`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 “创建DBParameter群组”。

CreateDBSnapshot

以下代码示例演示了如何使用 CreateDBSnapshot。

适用于 Java 的 SDK 2.x

> **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。

```
   // 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 的详细信息，请参阅DBSnapshot在 AWS SDK for Java 2.x API 参考中创建。

DeleteDBInstance

以下代码示例演示了如何使用 DeleteDBInstance。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅DBInstance《AWS SDK for Java 2.x API 参考》中的 “删除”。

`DeleteDBParameterGroup`

以下代码示例演示了如何使用 `DeleteDBParameterGroup`。

适用于 Java 的 SDK 2.x

> **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。

```
   // 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 的详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 “删除DBParameter群组”。

`DescribeAccountAttributes`

以下代码示例演示了如何使用 `DescribeAccountAttributes`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeAccountAttributes中的。

`DescribeDBEngineVersions`

以下代码示例演示了如何使用 `DescribeDBEngineVersions`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 “描述DBEngine版本”。

`DescribeDBInstances`

以下代码示例演示了如何使用 `DescribeDBInstances`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DBInstances中的描述。

`DescribeDBParameterGroups`

以下代码示例演示了如何使用 `DescribeDBParameterGroups`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 “描述DBParameter群组”。

`DescribeDBParameters`

以下代码示例演示了如何使用 `DescribeDBParameters`。

适用于 Java 的 SDK 2.x

> **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。

```
   // 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DBParameters中的描述。

`GenerateRDSAuthToken`

以下代码示例演示了如何使用 `GenerateRDSAuthToken`。

适用于 Java 的 SDK 2.x

> **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。

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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考中的生成RDSAuth令牌。

`ModifyDBInstance`

以下代码示例演示了如何使用 `ModifyDBInstance`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅《AWS SDK for Java 2.x API 参考DBInstance》中的 “修改”。

ModifyDBParameterGroup

以下代码示例演示了如何使用 ModifyDBParameterGroup。

适用于 Java 的 SDK 2.x

> **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。

```
   // 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 的详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的 “修改DBParameter群组”。

`RebootDBInstance`

以下代码示例演示了如何使用 `RebootDBInstance`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DBInstance中的重启。

场景

创建 Aurora Serverless 工作项跟踪器

以下代码示例演示如何创建 Web 应用程序，来跟踪 Amazon Aurora Serverless 数据库中的工作项，以及使用 Amazon Simple Email Service（Amazon SES）发送报告。

适用于 Java 的 SDK 2.x

展示如何创建 Web 应用程序来跟踪与报告存储与 Amazon RDS 数据库的工作项。

有关如何设置查询 Amazon Aurora Serverless 数据的 Spring REST API 以及如何让 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。

有关如何设置和运行使用 JDBC API 的示例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

- Aurora
- Amazon RDS
- Amazon RDS 数据服务
- Amazon SES

无服务器示例

使用 Lambda 函数连接到 Amazon RDS 数据库

以下代码示例显示如何实现连接到 RDS 数据库的 Lambda 函数。该函数发出一个简单的数据库请求并返回结果。

适用于 Java 的 SDK 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。

在 Lambda 函数中使用 Java 连接到 Amazon 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();
    }
}
```

使用适用于 Java 的 SDK 2.x 的 Amazon RDS 数据服务示例

以下代码示例向您展示了如何使用 AWS SDK for Java 2.x 与 Amazon RDS 数据服务配合使用来执行操作和实现常见场景。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 场景

场景

创建 Aurora Serverless 工作项跟踪器

以下代码示例演示如何创建 Web 应用程序，来跟踪 Amazon Aurora Serverless 数据库中的工作项，以及使用 Amazon Simple Email Service（Amazon SES）发送报告。

适用于 Java 的 SDK 2.x

展示如何创建 Web 应用程序来跟踪与报告存储与 Amazon RDS 数据库的工作项。

有关如何设置查询 Amazon Aurora Serverless 数据的 Spring REST API 以及如何让 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。

有关如何设置和运行使用 JDBC API 的示例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

- Aurora
- Amazon RDS
- Amazon RDS 数据服务
- Amazon SES

使用 SDK for 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。

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 基本功能
- 操作
- 场景

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。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeClusters中的。

基本功能

了解基本功能

以下代码示例展示了如何：

- 创建 Redshift 集群。
- 列出集群中的数据库。
- 创建名为 Movies 的文件。
- 填充 Movies 表。
- 按年份查询 Movies 表。
- 修改 Redshift 集群。
- 删除 Amazon Redshift 集群。

适用于 Java 的 SDK 2.x

> **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。

运行一个交互式场景，演示 Amazon Redshift 的功能。

```
import com.example.redshift.User;
import com.google.gson.Gson;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.redshift.model.ClusterAlreadyExistsException;
import software.amazon.awssdk.services.redshift.model.CreateClusterResponse;
import software.amazon.awssdk.services.redshift.model.DeleteClusterResponse;
import software.amazon.awssdk.services.redshift.model.ModifyClusterResponse;
import software.amazon.awssdk.services.redshift.model.RedshiftException;
import software.amazon.awssdk.services.redshiftdata.model.ExecuteStatementResponse;
import software.amazon.awssdk.services.redshiftdata.model.RedshiftDataException;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import software.amazon.awssdk.services.secretsmanager.SecretsManagerClient;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueRequest;
import software.amazon.awssdk.services.secretsmanager.model.GetSecretValueResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 *
 *  This example requires an AWS Secrets Manager secret that contains the
 *  database credentials. If you do not create a
 *  secret that specifies user name and password, this example will not work. For
 details, see:
 *
```

```
 *  https://docs.aws.amazon.com/secretsmanager/latest/userguide/integrating_how-
services-use-secrets_RS.html
 *
 This Java example performs these tasks:
 *
 * 1. Prompts the user for a unique cluster ID or use the default value.
 * 2. Creates a Redshift cluster with the specified or default cluster Id value.
 * 3. Waits until the Redshift cluster is available for use.
 * 4. Lists all databases using a pagination API call.
 * 5. Creates a table named "Movies" with fields ID, title, and year.
 * 6. Inserts a specified number of records into the "Movies" table by reading the
 Movies JSON file.
 * 7. Prompts the user for a movie release year.
 * 8. Runs a SQL query to retrieve movies released in the specified year.
 * 9. Modifies the Redshift cluster.
 * 10. Prompts the user for confirmation to delete the Redshift cluster.
 * 11. If confirmed, deletes the specified Redshift cluster.
 */

public class RedshiftScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final Logger logger =
 LoggerFactory.getLogger(RedshiftScenario.class);

    static RedshiftActions redshiftActions = new RedshiftActions();
    public static void main(String[] args) throws Exception {
        final String usage = """

            Usage:
                <jsonFilePath> <secretName>\s

            Where:
                jsonFilePath - The path to the Movies JSON file (you can locate that
 file in ../../../resources/sample_files/movies.json)
                secretName - The name of the secret that belongs to Secret Manager
 that stores the user name and password used in this scenario.
            """;

        if (args.length != 2) {
            logger.info(usage);
            return;
        }

        String jsonFilePath = args[0];
```

```
        String secretName = args[1];
        Scanner scanner = new Scanner(System.in);
        logger.info(DASHES);
        logger.info("Welcome to the Amazon Redshift SDK Basics scenario.");
        logger.info("""
            This Java program demonstrates how to interact with Amazon Redshift by
 using the AWS SDK for Java (v2).\s
            Amazon Redshift is a fully managed, petabyte-scale data warehouse
 service hosted in the cloud.

            The program's primary functionalities include cluster creation,
 verification of cluster readiness,\s
            list databases, table creation, data population within the table, and
 execution of SQL statements.
            Furthermore, it demonstrates the process of querying data from the Movie
 table.\s

            Upon completion of the program, all AWS resources are cleaned up.
            """);

        logger.info("Lets get started...");
        logger.info("""
            First, we will retrieve the user name and password from Secrets Manager.

            Using Amazon Secrets Manager to store Redshift credentials provides
 several security benefits.
            It allows you to securely store and manage sensitive information, such
 as passwords, API keys, and
            database credentials, without embedding them directly in your
 application code.

            More information can be found here:

            https://docs.aws.amazon.com/secretsmanager/latest/userguide/
integrating_how-services-use-secrets_RS.html
            """);
        Gson gson = new Gson();
        User user = gson.fromJson(String.valueOf(getSecretValues(secretName)),
 User.class);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        try {
            runScenario(user, scanner, jsonFilePath);
```

```
        } catch (RuntimeException e) {
            e.printStackTrace();
        } catch (Throwable e) {
            throw new RuntimeException(e);
        }
    }

    private static void runScenario(User user, Scanner scanner,  String
jsonFilePath) throws Throwable {
        String databaseName = "dev";
        System.out.println(DASHES);
        logger.info("Create a Redshift Cluster");
        logger.info("A Redshift cluster refers to the collection of computing
resources and storage that work together to process and analyze large volumes of
data.");
        logger.info("Enter a cluster id value or accept the default by hitting Enter
(default is redshift-cluster-movies): ");
        String userClusterId = scanner.nextLine();
        String clusterId = userClusterId.isEmpty() ? "redshift-cluster-movies" :
userClusterId;
        try {
            CompletableFuture<CreateClusterResponse> future =
redshiftActions.createClusterAsync(clusterId, user.getUserName(),
user.getUserPassword());
            CreateClusterResponse response = future.join();
            logger.info("Cluster successfully created. Cluster Identifier {} ",
response.cluster().clusterIdentifier());

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof ClusterAlreadyExistsException) {
                logger.info("The Cluster {} already exists. Moving on...",
clusterId);
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Wait until {} is available.", clusterId);
        waitForInputToContinue(scanner);
        try {
```

```
            CompletableFuture<Void> future =
 redshiftActions.waitForClusterReadyAsync(clusterId);
            future.join();
            logger.info("Cluster is ready!");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftException redshiftEx) {
                logger.info("Redshift error occurred: Error message: {}, Error code
 {}", redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        String databaseInfo = """
            When you created $clusteridD, the dev database is created by default and
 used in this scenario.\s

            To create a custom database, you need to have a CREATEDB privilege.\s
            For more information, see the documentation here: https://
docs.aws.amazon.com/redshift/latest/dg/r_CREATE_DATABASE.html.
           """.replace("$clusteridD", clusterId);

        logger.info(databaseInfo);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("List databases in {} ",clusterId);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
 redshiftActions.listAllDatabasesAsync(clusterId, user.getUserName(), "dev");
            future.join();
            logger.info("Databases listed successfully.");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
```

```
                logger.error("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.error("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Now you will create a table named Movies.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<ExecuteStatementResponse> future =
redshiftActions.createTableAsync(clusterId, databaseName, user.getUserName());
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Populate the Movies table using the Movies.json file.");
        logger.info("Specify the number of records you would like to add to the
Movies Table.");
        logger.info("Please enter a value between 50 and 200.");
        int numRecords;
        do {
            logger.info("Enter a value: ");
            while (!scanner.hasNextInt()) {
                logger.info("Invalid input. Please enter a value between 50 and
200.");
                logger.info("Enter a year: ");
                scanner.next();
            }
            numRecords = scanner.nextInt();
```

```
        } while (numRecords < 50 || numRecords > 200);
        try {
            redshiftActions.popTableAsync(clusterId, databaseName,
user.getUserName(), jsonFilePath, numRecords).join();  // Wait for the operation to
complete
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Query the Movies table by year. Enter a value between
2012-2014.");
        int movieYear;
        do {
            logger.info("Enter a year: ");
            while (!scanner.hasNextInt()) {
                logger.info("Invalid input. Please enter a valid year between 2012
and 2014.");
                logger.info("Enter a year: ");
                scanner.next();
            }
            movieYear = scanner.nextInt();
            scanner.nextLine();
        } while (movieYear < 2012 || movieYear > 2014);

        String id;
        try {
            CompletableFuture<String> future =
redshiftActions.queryMoviesByYearAsync(databaseName, user.getUserName(), movieYear,
clusterId);
            id = future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
```

```
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }

        logger.info("The identifier of the statement is " + id);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
redshiftActions.checkStatementAsync(id);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future = redshiftActions.getResultsAsync(id);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
```

```
        logger.info("Now you will modify the Redshift cluster.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<ModifyClusterResponse> future =
redshiftActions.modifyClusterAsync(clusterId);;
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof RedshiftDataException redshiftEx) {
                logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: {}", rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("Would you like to delete the Amazon Redshift cluster? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            logger.info("You selected to delete {} ", clusterId);
            waitForInputToContinue(scanner);
            try {
                CompletableFuture<DeleteClusterResponse> future =
redshiftActions.deleteRedshiftClusterAsync(clusterId);;
                future.join();

            } catch (RuntimeException rt) {
                Throwable cause = rt.getCause();
                if (cause instanceof RedshiftDataException redshiftEx) {
                    logger.info("Redshift Data error occurred: {} Error code: {}",
redshiftEx.getMessage(), redshiftEx.awsErrorDetails().errorCode());
                } else {
                    logger.info("An unexpected error occurred: {}",
rt.getMessage());
                }
                throw cause;
            }
        } else {
            logger.info("The {}  was not deleted", clusterId);
```

```
        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("This concludes the Amazon Redshift SDK Basics scenario.");
        logger.info(DASHES);
    }

    private static SecretsManagerClient getSecretClient() {
        Region region = Region.US_EAST_1;
        return SecretsManagerClient.builder()
            .region(region)
            .build();
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }

    // Get the Amazon Redshift credentials from AWS Secrets Manager.
    private static String getSecretValues(String secretName) {
        SecretsManagerClient secretClient = getSecretClient();
        GetSecretValueRequest valueRequest = GetSecretValueRequest.builder()
            .secretId(secretName)
            .build();

        GetSecretValueResponse valueResponse =
 secretClient.getSecretValue(valueRequest);
        return valueResponse.secretString();
    }
}
```

Amazon Redshift SDK 方法的封装类。

```
public class RedshiftActions {

    private static final Logger logger =
 LoggerFactory.getLogger(RedshiftActions.class);
    private static RedshiftDataAsyncClient redshiftDataAsyncClient;

    private static RedshiftAsyncClient redshiftAsyncClient;

    private static RedshiftAsyncClient getAsyncClient() {
        if (redshiftAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            redshiftAsyncClient = RedshiftAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return redshiftAsyncClient;
    }

    private static RedshiftDataAsyncClient getAsyncDataClient() {
        if (redshiftDataAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
```

```
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            redshiftDataAsyncClient = RedshiftDataAsyncClient.builder()
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return redshiftDataAsyncClient;
    }

    /**
     * Creates a new Amazon Redshift cluster asynchronously.
     * @param clusterId     the unique identifier for the cluster
     * @param username      the username for the administrative user
     * @param userPassword  the password for the administrative user
     * @return a CompletableFuture that represents the asynchronous operation of
creating the cluster
     * @throws RuntimeException if the cluster creation fails
     */
    public CompletableFuture<CreateClusterResponse> createClusterAsync(String
clusterId, String username, String userPassword) {
        CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .masterUsername(username)
            .masterUserPassword(userPassword)
            .nodeType("ra3.4xlarge")
            .publiclyAccessible(true)
            .numberOfNodes(2)
            .build();

        return getAsyncClient().createCluster(clusterRequest)
            .whenComplete((response, exception) -> {
                if (response != null) {
                    logger.info("Created cluster ");
                } else {
                    throw new RuntimeException("Failed to create cluster: " +
exception.getMessage(), exception);
```

```
                }
            });
    }

    /**
     * Waits asynchronously for the specified cluster to become available.
     * @param clusterId the identifier of the cluster to wait for
     * @return a {@link CompletableFuture} that completes when the cluster is ready
     */
    public CompletableFuture<Void> waitForClusterReadyAsync(String clusterId) {
        DescribeClustersRequest clustersRequest = DescribeClustersRequest.builder()
            .clusterIdentifier(clusterId)
            .build();

        logger.info("Waiting for cluster to become available. This may take a few
minutes.");
        long startTime = System.currentTimeMillis();

        // Recursive method to poll the cluster status.
        return checkClusterStatusAsync(clustersRequest, startTime);
    }

    private CompletableFuture<Void> checkClusterStatusAsync(DescribeClustersRequest
clustersRequest, long startTime) {
        return getAsyncClient().describeClusters(clustersRequest)
            .thenCompose(clusterResponse -> {
                List<Cluster> clusterList = clusterResponse.clusters();
                boolean clusterReady = false;
                for (Cluster cluster : clusterList) {
                    if ("available".equals(cluster.clusterStatus())) {
                        clusterReady = true;
                        break;
                    }
                }

                if (clusterReady) {
                    logger.info(String.format("Cluster is available!"));
                    return CompletableFuture.completedFuture(null);
                } else {
                    long elapsedTimeMillis = System.currentTimeMillis() - startTime;
                    long elapsedSeconds = elapsedTimeMillis / 1000;
                    long minutes = elapsedSeconds / 60;
                    long seconds = elapsedSeconds % 60;
```

```
                    System.out.printf("\rElapsed Time: %02d:%02d - Waiting for
cluster...", minutes, seconds);
                    System.out.flush();

                    // Wait 1 second before the next status check
                    return CompletableFuture.runAsync(() -> {
                        try {
                            TimeUnit.SECONDS.sleep(1);
                        } catch (InterruptedException e) {
                            throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
                        }
                    }).thenCompose(ignored ->
checkClusterStatusAsync(clustersRequest, startTime));
                }
            }).exceptionally(exception -> {
                throw new RuntimeException("Failed to get cluster status: " +
exception.getMessage(), exception);
            });
    }

    /**
     * Lists all databases asynchronously for the specified cluster, database user,
and database.
     * @param clusterId the identifier of the cluster to list databases for
     * @param dbUser the database user to use for the list databases request
     * @param database the database to list databases for
     * @return a {@link CompletableFuture} that completes when the database listing
is complete, or throws a {@link RuntimeException} if there was an error
     */
    public CompletableFuture<Void> listAllDatabasesAsync(String clusterId, String
dbUser, String database) {
        ListDatabasesRequest databasesRequest = ListDatabasesRequest.builder()
            .clusterIdentifier(clusterId)
            .dbUser(dbUser)
            .database(database)
            .build();

        // Asynchronous paginator for listing databases.
        ListDatabasesPublisher databasesPaginator =
getAsyncDataClient().listDatabasesPaginator(databasesRequest);
        CompletableFuture<Void> future = databasesPaginator.subscribe(response -> {
            response.databases().forEach(db -> {
                logger.info("The database name is {} ", db);
```

```
            });
        });

        // Return the future for asynchronous handling.
        return future.exceptionally(exception -> {
            throw new RuntimeException("Failed to list databases: " +
exception.getMessage(), exception);
        });
    }

    /**
     * Creates an asynchronous task to execute a SQL statement for creating a new
table.
     *
     * @param clusterId    the identifier of the Amazon Redshift cluster
     * @param databaseName the name of the database to create the table in
     * @param userName     the username to use for the database connection
     * @return a {@link CompletableFuture} that completes with the result of the SQL
statement execution
     * @throws RuntimeException if there is an error creating the table
     */
    public CompletableFuture<ExecuteStatementResponse> createTableAsync(String
clusterId, String databaseName, String userName) {
        ExecuteStatementRequest createTableRequest =
ExecuteStatementRequest.builder()
            .clusterIdentifier(clusterId)
            .dbUser(userName)
            .database(databaseName)
            .sql("CREATE TABLE Movies (" +
                "id INT PRIMARY KEY, " +
                "title VARCHAR(100), " +
                "year INT)")
            .build();

        return getAsyncDataClient().executeStatement(createTableRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    throw new RuntimeException("Error creating table: " +
exception.getMessage(), exception);
                } else {
                    logger.info("Table created: Movies");
                }
            });
    }
```

```
   /**
    * Asynchronously pops a table from a JSON file.
    *
    * @param clusterId   the ID of the cluster
    * @param databaseName the name of the database
    * @param userName    the username
    * @param fileName    the name of the JSON file
    * @param number      the number of records to process
    * @return a CompletableFuture that completes with the number of records added
to the Movies table
    */
   public CompletableFuture<Integer> popTableAsync(String clusterId, String
databaseName, String userName, String fileName, int number) {
       return CompletableFuture.supplyAsync(() -> {
               try {
                   JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                   JsonNode rootNode = new ObjectMapper().readTree(parser);
                   Iterator<JsonNode> iter = rootNode.iterator();
                   return iter;
               } catch (IOException e) {
                   throw new RuntimeException("Failed to read or parse JSON file: "
+ e.getMessage(), e);
               }
           }).thenCompose(iter -> processNodesAsync(clusterId, databaseName,
userName, iter, number))
           .whenComplete((result, exception) -> {
               if (exception != null) {
                   logger.info("Error {} ", exception.getMessage());
               } else {
                   logger.info("{} records were added to the Movies table." ,
result);
               }
           });
   }

   private CompletableFuture<Integer> processNodesAsync(String clusterId, String
databaseName, String userName, Iterator<JsonNode> iter, int number) {
       return CompletableFuture.supplyAsync(() -> {
           int t = 0;
           try {
               while (iter.hasNext()) {
                   if (t == number)
```

```
                    break;
                JsonNode currentNode = iter.next();
                int year = currentNode.get("year").asInt();
                String title = currentNode.get("title").asText();

                // Use SqlParameter to avoid SQL injection.
                List<SqlParameter> parameterList = new ArrayList<>();
                String sqlStatement = "INSERT INTO Movies
VALUES( :id , :title, :year);";
                SqlParameter idParam = SqlParameter.builder()
                    .name("id")
                    .value(String.valueOf(t))
                    .build();

                SqlParameter titleParam = SqlParameter.builder()
                    .name("title")
                    .value(title)
                    .build();

                SqlParameter yearParam = SqlParameter.builder()
                    .name("year")
                    .value(String.valueOf(year))
                    .build();
                parameterList.add(idParam);
                parameterList.add(titleParam);
                parameterList.add(yearParam);

                ExecuteStatementRequest insertStatementRequest =
ExecuteStatementRequest.builder()
                    .clusterIdentifier(clusterId)
                    .sql(sqlStatement)
                    .database(databaseName)
                    .dbUser(userName)
                    .parameters(parameterList)
                    .build();

                getAsyncDataClient().executeStatement(insertStatementRequest);
                logger.info("Inserted: " + title + " (" + year + ")");
                t++;
            }
        } catch (RedshiftDataException e) {
            throw new RuntimeException("Error inserting data: " +
e.getMessage(), e);
        }
```

```
            return t;
        });
    }

    /**
     * Checks the status of an SQL statement asynchronously and handles the
completion of the statement.
     *
     * @param sqlId the ID of the SQL statement to check
     * @return a {@link CompletableFuture} that completes when the SQL statement's
status is either "FINISHED" or "FAILED"
     */
    public CompletableFuture<Void> checkStatementAsync(String sqlId) {
        DescribeStatementRequest statementRequest =
DescribeStatementRequest.builder()
            .id(sqlId)
            .build();

        return getAsyncDataClient().describeStatement(statementRequest)
            .thenCompose(response -> {
                String status = response.statusAsString();
                logger.info("... Status: {} ", status);

                if ("FAILED".equals(status)) {
                    throw new RuntimeException("The Query Failed. Ending program");
                } else if ("FINISHED".equals(status)) {
                    return CompletableFuture.completedFuture(null);
                } else {
                    // Sleep for 1 second and recheck status
                    return CompletableFuture.runAsync(() -> {
                        try {
                            TimeUnit.SECONDS.sleep(1);
                        } catch (InterruptedException e) {
                            throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
                        }
                    }).thenCompose(ignore -> checkStatementAsync(sqlId)); //
Recursively call until status is FINISHED or FAILED
                }
            }).whenComplete((result, exception) -> {
                if (exception != null) {
                    // Handle exceptions
                    logger.info("Error: {} ", exception.getMessage());
                } else {
```

```
                    logger.info("The statement is finished!");
                }
            });
    }

    /**
     * Asynchronously retrieves the results of a statement execution.
     *
     * @param statementId the ID of the statement for which to retrieve the results
     * @return a {@link CompletableFuture} that completes when the statement result
has been processed
     */
    public CompletableFuture<Void> getResultsAsync(String statementId) {
        GetStatementResultRequest resultRequest =
GetStatementResultRequest.builder()
            .id(statementId)
            .build();

        return getAsyncDataClient().getStatementResult(resultRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    logger.info("Error getting statement result {} ",
exception.getMessage());
                    throw new RuntimeException("Error getting statement result: " +
exception.getMessage(), exception);
                }

                // Extract and print the field values using streams if the response
is valid.
                response.records().stream()
                    .flatMap(List::stream)
                    .map(Field::stringValue)
                    .filter(value -> value != null)
                    .forEach(value -> System.out.println("The Movie title field is "
+ value));

                return response;
            }).thenAccept(response -> {
                // Optionally add more logic here if needed after handling the
response
            });
    }
```

```
    /**
     * Asynchronously queries movies by a given year from a Redshift database.
     *
     * @param database    the name of the database to query
     * @param dbUser      the user to connect to the database with
     * @param year        the year to filter the movies by
     * @param clusterId   the identifier of the Redshift cluster to connect to
     * @return a {@link CompletableFuture} containing the response ID of the
executed SQL statement
     */
    public CompletableFuture<String> queryMoviesByYearAsync(String database,
                                                            String dbUser,
                                                            int year,
                                                            String clusterId)
{

        String sqlStatement = "SELECT * FROM Movies WHERE year = :year";
        SqlParameter yearParam = SqlParameter.builder()
            .name("year")
            .value(String.valueOf(year))
            .build();

        ExecuteStatementRequest statementRequest = ExecuteStatementRequest.builder()
            .clusterIdentifier(clusterId)
            .database(database)
            .dbUser(dbUser)
            .parameters(yearParam)
            .sql(sqlStatement)
            .build();

        return CompletableFuture.supplyAsync(() -> {
            try {
                ExecuteStatementResponse response =
getAsyncDataClient().executeStatement(statementRequest).join(); // Use join() to
wait for the result
                return response.id();
            } catch (RedshiftDataException e) {
                throw new RuntimeException("Error executing statement: " +
e.getMessage(), e);
            }
        }).exceptionally(exception -> {
            logger.info("Error: {}", exception.getMessage());
            return "";
        });
```

```
    }

    /**
     * Modifies an Amazon Redshift cluster asynchronously.
     *
     * @param clusterId the identifier of the cluster to be modified
     * @return a {@link CompletableFuture} that completes when the cluster
modification is complete
     */
    public CompletableFuture<ModifyClusterResponse> modifyClusterAsync(String
clusterId) {
        ModifyClusterRequest modifyClusterRequest = ModifyClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .preferredMaintenanceWindow("wed:07:30-wed:08:00")
            .build();

        return getAsyncClient().modifyCluster(modifyClusterRequest)
            .whenComplete((clusterResponse, exception) -> {
                if (exception != null) {
                    if (exception.getCause() instanceof RedshiftException) {
                        logger.info("Error: {} ", exception.getMessage());
                    } else {
                        logger.info("Unexpected error: {} ",
exception.getMessage());
                    }
                } else {
                    logger.info("The modified cluster was successfully modified and
has "
                        + clusterResponse.cluster().preferredMaintenanceWindow() + "
as the maintenance window");
                }
            });
    }

    /**
     * Deletes a Redshift cluster asynchronously.
     *
     * @param clusterId the identifier of the Redshift cluster to be deleted
     * @return a {@link CompletableFuture} that represents the asynchronous
operation of deleting the Redshift cluster
     */
    public CompletableFuture<DeleteClusterResponse>
deleteRedshiftClusterAsync(String clusterId) {
        DeleteClusterRequest deleteClusterRequest = DeleteClusterRequest.builder()
```

```
            .clusterIdentifier(clusterId)
            .skipFinalClusterSnapshot(true)
            .build();

        return getAsyncClient().deleteCluster(deleteClusterRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    // Handle exceptions
                    if (exception.getCause() instanceof RedshiftException) {
                        logger.info("Error: {}", exception.getMessage());
                    } else {
                        logger.info("Unexpected error: {}", exception.getMessage());
                    }
                } else {
                    // Handle successful response
                    logger.info("The status is {}",
 response.cluster().clusterStatus());
                }
            });
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - CreateCluster
  - DescribeClusters
  - DescribeStatement
  - ExecuteStatement
  - GetStatementResult
  - ListDatabasesPaginator
  - ModifyCluster

操作

`CreateCluster`

以下代码示例演示了如何使用 `CreateCluster`。

适用于 Java 的 SDK 2.x

> **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。

创建集群。

```
   /**
    * Creates a new Amazon Redshift cluster asynchronously.
    * @param clusterId     the unique identifier for the cluster
    * @param username      the username for the administrative user
    * @param userPassword  the password for the administrative user
    * @return a CompletableFuture that represents the asynchronous operation of
creating the cluster
    * @throws RuntimeException if the cluster creation fails
    */
   public CompletableFuture<CreateClusterResponse> createClusterAsync(String
clusterId, String username, String userPassword) {
       CreateClusterRequest clusterRequest = CreateClusterRequest.builder()
           .clusterIdentifier(clusterId)
           .masterUsername(username)
           .masterUserPassword(userPassword)
           .nodeType("ra3.4xlarge")
           .publiclyAccessible(true)
           .numberOfNodes(2)
           .build();

       return getAsyncClient().createCluster(clusterRequest)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   logger.info("Created cluster ");
               } else {
                   throw new RuntimeException("Failed to create cluster: " +
exception.getMessage(), exception);
               }
           });
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateCluster中的。

`DeleteCluster`

以下代码示例演示了如何使用 `DeleteCluster`。

适用于 Java 的 SDK 2.x

> **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。

请删除集群。

```
   /**
    * Deletes a Redshift cluster asynchronously.
    *
    * @param clusterId the identifier of the Redshift cluster to be deleted
    * @return a {@link CompletableFuture} that represents the asynchronous
operation of deleting the Redshift cluster
    */
   public CompletableFuture<DeleteClusterResponse>
deleteRedshiftClusterAsync(String clusterId) {
       DeleteClusterRequest deleteClusterRequest = DeleteClusterRequest.builder()
           .clusterIdentifier(clusterId)
           .skipFinalClusterSnapshot(true)
           .build();

       return getAsyncClient().deleteCluster(deleteClusterRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   // Handle exceptions
                   if (exception.getCause() instanceof RedshiftException) {
                       logger.info("Error: {}", exception.getMessage());
                   } else {
                       logger.info("Unexpected error: {}", exception.getMessage());
                   }
               } else {
                   // Handle successful response
```

```
                    logger.info("The status is {}",
response.cluster().clusterStatus());
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteCluster中的。

`DescribeClusters`

以下代码示例演示了如何使用 `DescribeClusters`。

适用于 Java 的 SDK 2.x

> **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。

描述集群。

```
    /**
     * Waits asynchronously for the specified cluster to become available.
     * @param clusterId the identifier of the cluster to wait for
     * @return a {@link CompletableFuture} that completes when the cluster is ready
     */
    public CompletableFuture<Void> waitForClusterReadyAsync(String clusterId) {
        DescribeClustersRequest clustersRequest = DescribeClustersRequest.builder()
            .clusterIdentifier(clusterId)
            .build();

        logger.info("Waiting for cluster to become available. This may take a few
minutes.");
        long startTime = System.currentTimeMillis();

        // Recursive method to poll the cluster status.
        return checkClusterStatusAsync(clustersRequest, startTime);
    }
```

```
   private CompletableFuture<Void> checkClusterStatusAsync(DescribeClustersRequest
clustersRequest, long startTime) {
       return getAsyncClient().describeClusters(clustersRequest)
           .thenCompose(clusterResponse -> {
               List<Cluster> clusterList = clusterResponse.clusters();
               boolean clusterReady = false;
               for (Cluster cluster : clusterList) {
                   if ("available".equals(cluster.clusterStatus())) {
                       clusterReady = true;
                       break;
                   }
               }

               if (clusterReady) {
                   logger.info(String.format("Cluster is available!"));
                   return CompletableFuture.completedFuture(null);
               } else {
                   long elapsedTimeMillis = System.currentTimeMillis() - startTime;
                   long elapsedSeconds = elapsedTimeMillis / 1000;
                   long minutes = elapsedSeconds / 60;
                   long seconds = elapsedSeconds % 60;
                   System.out.printf("\rElapsed Time: %02d:%02d - Waiting for
cluster...", minutes, seconds);
                   System.out.flush();

                   // Wait 1 second before the next status check
                   return CompletableFuture.runAsync(() -> {
                       try {
                           TimeUnit.SECONDS.sleep(1);
                       } catch (InterruptedException e) {
                           throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
                       }
                   }).thenCompose(ignored ->
checkClusterStatusAsync(clustersRequest, startTime));
               }
           }).exceptionally(exception -> {
               throw new RuntimeException("Failed to get cluster status: " +
exception.getMessage(), exception);
           });
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeClusters中的。

DescribeStatement

以下代码示例演示了如何使用 DescribeStatement。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Checks the status of an SQL statement asynchronously and handles the
completion of the statement.
     *
     * @param sqlId the ID of the SQL statement to check
     * @return a {@link CompletableFuture} that completes when the SQL statement's
status is either "FINISHED" or "FAILED"
     */
    public CompletableFuture<Void> checkStatementAsync(String sqlId) {
        DescribeStatementRequest statementRequest =
DescribeStatementRequest.builder()
            .id(sqlId)
            .build();

        return getAsyncDataClient().describeStatement(statementRequest)
            .thenCompose(response -> {
                String status = response.statusAsString();
                logger.info("... Status: {} ", status);

                if ("FAILED".equals(status)) {
                    throw new RuntimeException("The Query Failed. Ending program");
                } else if ("FINISHED".equals(status)) {
                    return CompletableFuture.completedFuture(null);
                } else {
                    // Sleep for 1 second and recheck status
                    return CompletableFuture.runAsync(() -> {
                        try {
                            TimeUnit.SECONDS.sleep(1);
                        } catch (InterruptedException e) {
                            throw new RuntimeException("Error during sleep: " +
e.getMessage(), e);
```

```
                    }
                }).thenCompose(ignore -> checkStatementAsync(sqlId)); //
Recursively call until status is FINISHED or FAILED
            }
        }).whenComplete((result, exception) -> {
            if (exception != null) {
                // Handle exceptions
                logger.info("Error: {} ", exception.getMessage());
            } else {
                logger.info("The statement is finished!");
            }
        });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeStatement中的。

`ExecuteStatement`

以下代码示例演示了如何使用 `ExecuteStatement`。

适用于 Java 的 SDK 2.x

> **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。

执行 SQL 语句来创建数据库表。

```
    /**
     * Creates an asynchronous task to execute a SQL statement for creating a new
table.
     *
     * @param clusterId    the identifier of the Amazon Redshift cluster
     * @param databaseName the name of the database to create the table in
     * @param userName     the username to use for the database connection
     * @return a {@link CompletableFuture} that completes with the result of the SQL
statement execution
     * @throws RuntimeException if there is an error creating the table
```

```
    */
   public CompletableFuture<ExecuteStatementResponse> createTableAsync(String
clusterId, String databaseName, String userName) {
       ExecuteStatementRequest createTableRequest =
ExecuteStatementRequest.builder()
           .clusterIdentifier(clusterId)
           .dbUser(userName)
           .database(databaseName)
           .sql("CREATE TABLE Movies (" +
               "id INT PRIMARY KEY, " +
               "title VARCHAR(100), " +
               "year INT)")
           .build();

       return getAsyncDataClient().executeStatement(createTableRequest)
           .whenComplete((response, exception) -> {
               if (exception != null) {
                   throw new RuntimeException("Error creating table: " +
exception.getMessage(), exception);
               } else {
                   logger.info("Table created: Movies");
               }
           });
   }
```

执行 SQL 语句将数据插入数据库表。

```
   /**
    * Asynchronously pops a table from a JSON file.
    *
    * @param clusterId   the ID of the cluster
    * @param databaseName the name of the database
    * @param userName    the username
    * @param fileName    the name of the JSON file
    * @param number      the number of records to process
    * @return a CompletableFuture that completes with the number of records added
to the Movies table
    */
   public CompletableFuture<Integer> popTableAsync(String clusterId, String
databaseName, String userName, String fileName, int number) {
       return CompletableFuture.supplyAsync(() -> {
               try {
```

```
                    JsonParser parser = new JsonFactory().createParser(new
File(fileName));
                    JsonNode rootNode = new ObjectMapper().readTree(parser);
                    Iterator<JsonNode> iter = rootNode.iterator();
                    return iter;
                } catch (IOException e) {
                    throw new RuntimeException("Failed to read or parse JSON file: "
+ e.getMessage(), e);
                }
            }).thenCompose(iter -> processNodesAsync(clusterId, databaseName,
userName, iter, number))
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    logger.info("Error {} ", exception.getMessage());
                } else {
                    logger.info("{} records were added to the Movies table." ,
result);
                }
            });
    }

    private CompletableFuture<Integer> processNodesAsync(String clusterId, String
databaseName, String userName, Iterator<JsonNode> iter, int number) {
        return CompletableFuture.supplyAsync(() -> {
            int t = 0;
            try {
                while (iter.hasNext()) {
                    if (t == number)
                        break;
                    JsonNode currentNode = iter.next();
                    int year = currentNode.get("year").asInt();
                    String title = currentNode.get("title").asText();

                    // Use SqlParameter to avoid SQL injection.
                    List<SqlParameter> parameterList = new ArrayList<>();
                    String sqlStatement = "INSERT INTO Movies
VALUES( :id , :title, :year);";
                    SqlParameter idParam = SqlParameter.builder()
                        .name("id")
                        .value(String.valueOf(t))
                        .build();

                    SqlParameter titleParam = SqlParameter.builder()
                        .name("title")
```

```
                        .value(title)
                        .build();

                SqlParameter yearParam = SqlParameter.builder()
                        .name("year")
                        .value(String.valueOf(year))
                        .build();
                parameterList.add(idParam);
                parameterList.add(titleParam);
                parameterList.add(yearParam);

                ExecuteStatementRequest insertStatementRequest =
ExecuteStatementRequest.builder()
                        .clusterIdentifier(clusterId)
                        .sql(sqlStatement)
                        .database(databaseName)
                        .dbUser(userName)
                        .parameters(parameterList)
                        .build();

                getAsyncDataClient().executeStatement(insertStatementRequest);
                logger.info("Inserted: " + title + " (" + year + ")");
                t++;
            }
        } catch (RedshiftDataException e) {
            throw new RuntimeException("Error inserting data: " +
e.getMessage(), e);
        }
        return t;
    });
  }
```

执行 SQL 语句来查询数据库表。

```
  /**
   * Asynchronously queries movies by a given year from a Redshift database.
   *
   * @param database    the name of the database to query
   * @param dbUser      the user to connect to the database with
   * @param year        the year to filter the movies by
   * @param clusterId   the identifier of the Redshift cluster to connect to
```

```
     * @return a {@link CompletableFuture} containing the response ID of the
executed SQL statement
     */
    public CompletableFuture<String> queryMoviesByYearAsync(String database,
                                                            String dbUser,
                                                            int year,
                                                            String clusterId)
{

        String sqlStatement = "SELECT * FROM Movies WHERE year = :year";
        SqlParameter yearParam = SqlParameter.builder()
            .name("year")
            .value(String.valueOf(year))
            .build();

        ExecuteStatementRequest statementRequest = ExecuteStatementRequest.builder()
            .clusterIdentifier(clusterId)
            .database(database)
            .dbUser(dbUser)
            .parameters(yearParam)
            .sql(sqlStatement)
            .build();

        return CompletableFuture.supplyAsync(() -> {
            try {
                ExecuteStatementResponse response =
getAsyncDataClient().executeStatement(statementRequest).join(); // Use join() to
wait for the result
                return response.id();
            } catch (RedshiftDataException e) {
                throw new RuntimeException("Error executing statement: " +
e.getMessage(), e);
            }
        }).exceptionally(exception -> {
            logger.info("Error: {}", exception.getMessage());
            return "";
        });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ExecuteStatement](#)中的。

GetStatementResult

以下代码示例演示了如何使用 GetStatementResult。

适用于 Java 的 SDK 2.x

> **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。

检查语句结果。

```
    /**
     * Asynchronously retrieves the results of a statement execution.
     *
     * @param statementId the ID of the statement for which to retrieve the results
     * @return a {@link CompletableFuture} that completes when the statement result
has been processed
     */
    public CompletableFuture<Void> getResultsAsync(String statementId) {
        GetStatementResultRequest resultRequest =
GetStatementResultRequest.builder()
            .id(statementId)
            .build();

        return getAsyncDataClient().getStatementResult(resultRequest)
            .handle((response, exception) -> {
                if (exception != null) {
                    logger.info("Error getting statement result {} ",
exception.getMessage());
                    throw new RuntimeException("Error getting statement result: " +
exception.getMessage(), exception);
                }

                // Extract and print the field values using streams if the response
is valid.
                response.records().stream()
                    .flatMap(List::stream)
                    .map(Field::stringValue)
                    .filter(value -> value != null)
```

```
                    .forEach(value -> System.out.println("The Movie title field is "
+ value));

                return response;
            }).thenAccept(response -> {
                // Optionally add more logic here if needed after handling the
response
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetStatementResult中的。

`ListDatabases`

以下代码示例演示了如何使用 `ListDatabases`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Lists all databases asynchronously for the specified cluster, database user,
and database.
     * @param clusterId the identifier of the cluster to list databases for
     * @param dbUser the database user to use for the list databases request
     * @param database the database to list databases for
     * @return a {@link CompletableFuture} that completes when the database listing
is complete, or throws a {@link RuntimeException} if there was an error
     */
    public CompletableFuture<Void> listAllDatabasesAsync(String clusterId, String
dbUser, String database) {
        ListDatabasesRequest databasesRequest = ListDatabasesRequest.builder()
            .clusterIdentifier(clusterId)
            .dbUser(dbUser)
            .database(database)
            .build();
```

```
        // Asynchronous paginator for listing databases.
        ListDatabasesPublisher databasesPaginator =
getAsyncDataClient().listDatabasesPaginator(databasesRequest);
        CompletableFuture<Void> future = databasesPaginator.subscribe(response -> {
            response.databases().forEach(db -> {
                logger.info("The database name is {} ", db);
            });
        });

        // Return the future for asynchronous handling.
        return future.exceptionally(exception -> {
            throw new RuntimeException("Failed to list databases: " +
exception.getMessage(), exception);
        });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListDatabases中的。

`ModifyCluster`

以下代码示例演示了如何使用 `ModifyCluster`。

适用于 Java 的 SDK 2.x

> **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。

修改集群。

```
    /**
     * Modifies an Amazon Redshift cluster asynchronously.
     *
     * @param clusterId the identifier of the cluster to be modified
     * @return a {@link CompletableFuture} that completes when the cluster
modification is complete
     */
    public CompletableFuture<ModifyClusterResponse> modifyClusterAsync(String
clusterId) {
```

```
        ModifyClusterRequest modifyClusterRequest = ModifyClusterRequest.builder()
            .clusterIdentifier(clusterId)
            .preferredMaintenanceWindow("wed:07:30-wed:08:00")
            .build();

        return getAsyncClient().modifyCluster(modifyClusterRequest)
            .whenComplete((clusterResponse, exception) -> {
                if (exception != null) {
                    if (exception.getCause() instanceof RedshiftException) {
                        logger.info("Error: {} ", exception.getMessage());
                    } else {
                        logger.info("Unexpected error: {} ",
exception.getMessage());
                    }
                } else {
                    logger.info("The modified cluster was successfully modified and
has "
                        + clusterResponse.cluster().preferredMaintenanceWindow() + "
as the maintenance window");
                }
            });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ModifyCluster中的。

场景

创建 Web 应用程序来跟踪 Amazon Redshift 数据

以下代码示例演示如何使用 Amazon Redshift 数据库创建用于跟踪和报告工作项的 Web 应用程序。

适用于 Java 的 SDK 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 数据的 Spring REST API 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
- Amazon SES

使用 SDK for Java 2.x 的 Amazon Rekognition 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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从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`

以下代码示例演示了如何使用 `CompareFaces`。

有关更多信息，请参阅比较图像中的人脸。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;
import software.amazon.awssdk.core.SdkBytes;
```

```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CompareFaces {
    public static void main(String[] args) {
        final String usage = """
            Usage: <bucketName> <sourceKey> <targetKey>

            Where:
                bucketName - The name of the S3 bucket where the images are stored.
                sourceKey  - The S3 key (file name) for the source image.
                targetKey  - The S3 key (file name) for the target image.
           """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceKey = args[1];
        String targetKey = args[2];

        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();
        compareTwoFaces(rekClient, bucketName, sourceKey, targetKey);
    }

    /**
     * Compares two faces from images stored in an Amazon S3 bucket using AWS
 Rekognition.
     *
```

```
    * <p>This method takes two image keys from an S3 bucket and compares the faces
within them.
    * It prints out the confidence level of matched faces and reports the number of
unmatched faces.</p>
    *
    * @param rekClient   The {@link RekognitionClient} used to call AWS
Rekognition.
    * @param bucketName  The name of the S3 bucket containing the images.
    * @param sourceKey   The object key (file path) for the source image in the S3
bucket.
    * @param targetKey   The object key (file path) for the target image in the S3
bucket.
    * @throws RuntimeException If the Rekognition service returns an error.
    */
   public static void compareTwoFaces(RekognitionClient rekClient, String
bucketName, String sourceKey, String targetKey) {
       try {
           Float similarityThreshold = 70F;
           S3Object s3ObjectSource = S3Object.builder()
                   .bucket(bucketName)
                   .name(sourceKey)
                   .build();

           Image sourceImage = Image.builder()
                   .s3Object(s3ObjectSource)
                   .build();

           S3Object s3ObjectTarget = S3Object.builder()
                   .bucket(bucketName)
                   .name(targetKey)
                   .build();

           Image targetImage = Image.builder()
                   .s3Object(s3ObjectTarget)
                   .build();

           CompareFacesRequest facesRequest = CompareFacesRequest.builder()
                   .sourceImage(sourceImage)
                   .targetImage(targetImage)
                   .similarityThreshold(similarityThreshold)
                   .build();

           // Compare the two images.
```

```
            CompareFacesResponse compareFacesResult =
 rekClient.compareFaces(facesRequest);
            List<CompareFacesMatch> faceDetails = compareFacesResult.faceMatches();

            for (CompareFacesMatch match : faceDetails) {
                ComparedFace face = match.face();
                BoundingBox position = face.boundingBox();
                System.out.println("Face at " + position.left().toString()
                        + " " + position.top()
                        + " matches with " + face.confidence().toString()
                        + "% confidence.");
            }

            List<ComparedFace> unmatchedFaces = compareFacesResult.unmatchedFaces();
            System.out.println("There were " + unmatchedFaces.size() + " face(s)
 that did not match.");

        } catch (RekognitionException e) {
            System.err.println("Error comparing faces: " +
 e.awsErrorDetails().errorMessage());
            throw new RuntimeException(e);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CompareFaces中的。

`CreateCollection`

以下代码示例演示了如何使用 `CreateCollection`。

有关更多信息，请参阅创建集合。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.CreateCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateCollection {
    public static void main(String[] args) {
        final String usage = """

            Usage: <collectionName>\s

            Where:
                collectionName - The name of the collection.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Creating collection: " + collectionId);
        createMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    /**
     * Creates a new Amazon Rekognition collection.
     *
```

```
     * @param rekClient    the Amazon Rekognition client used to interact with the
 Rekognition service
     * @param collectionId the unique identifier for the collection to be created
     */
    public static void createMyCollection(RekognitionClient rekClient, String
 collectionId) {
        try {
            CreateCollectionRequest collectionRequest =
 CreateCollectionRequest.builder()
                    .collectionId(collectionId)
                    .build();

            CreateCollectionResponse collectionResponse =
 rekClient.createCollection(collectionRequest);
            System.out.println("CollectionArn: " +
 collectionResponse.collectionArn());
            System.out.println("Status code: " +
 collectionResponse.statusCode().toString());

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateCollection中的。

DeleteCollection

以下代码示例演示了如何使用 DeleteCollection。

有关更多信息，请参阅删除集合。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.DeleteCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteCollection {
    public static void main(String[] args) {
        final String usage = """
            Usage: <collectionId>\s

            Where:
                collectionId - The id of the collection to delete.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Deleting collection: " + collectionId);
        deleteMyCollection(rekClient, collectionId);
        rekClient.close();
    }

    /**
     * Deletes an Amazon Rekognition collection.
     *
```

```
     * @param rekClient      An instance of the {@link RekognitionClient} class,
 which is used to interact with the Amazon Rekognition service.
     * @param collectionId   The ID of the collection to be deleted.
     */
    public static void deleteMyCollection(RekognitionClient rekClient, String
 collectionId) {
        try {
            DeleteCollectionRequest deleteCollectionRequest =
 DeleteCollectionRequest.builder()
                    .collectionId(collectionId)
                    .build();

            DeleteCollectionResponse deleteCollectionResponse =
 rekClient.deleteCollection(deleteCollectionRequest);
            System.out.println(collectionId + ": " +
 deleteCollectionResponse.statusCode().toString());

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteCollection中的。

`DeleteFaces`

以下代码示例演示了如何使用 `DeleteFaces`。

有关更多信息，请参阅从集合中删除人脸。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteFacesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DeleteFacesFromCollection {
    public static void main(String[] args) {
        final String usage = """
            Usage: <collectionId> <faceId>\s

            Where:
                collectionId - The id of the collection from which faces are
 deleted.\s
                faceId - The id of the face to delete.\s
           """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String faceId = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Deleting collection: " + collectionId);
        deleteFacesCollection(rekClient, collectionId, faceId);
        rekClient.close();
    }

    /**
     * Deletes a face from the specified Amazon Rekognition collection.
     *
     * @param rekClient     an instance of the Amazon Rekognition client
```

```
     * @param collectionId  the ID of the collection from which the face should be
 deleted
     * @param faceId        the ID of the face to be deleted
     * @throws RekognitionException if an error occurs while deleting the face
     */
    public static void deleteFacesCollection(RekognitionClient rekClient,
            String collectionId,
            String faceId) {

        try {
            DeleteFacesRequest deleteFacesRequest = DeleteFacesRequest.builder()
                    .collectionId(collectionId)
                    .faceIds(faceId)
                    .build();

            rekClient.deleteFaces(deleteFacesRequest);
            System.out.println("The face was deleted from the collection.");

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteFaces中的。

`DescribeCollection`

以下代码示例演示了如何使用 `DescribeCollection`。

有关更多信息，请参阅描述集合。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DescribeCollectionRequest;
import software.amazon.awssdk.services.rekognition.model.DescribeCollectionResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DescribeCollection {
    public static void main(String[] args) {
        final String usage = """
            Usage:    <collectionName>

            Where:
                collectionName - The name of the Amazon Rekognition collection.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionName = args[0];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        describeColl(rekClient, collectionName);
        rekClient.close();
    }

    /**
     * Describes an Amazon Rekognition collection.
     *
     * @param rekClient         The Amazon Rekognition client used to make the
 request.
```

```
     * @param collectionName    The name of the collection to describe.
     *
     * @throws RekognitionException If an error occurs while describing the
 collection.
     */
    public static void describeColl(RekognitionClient rekClient, String
 collectionName) {
        try {
            DescribeCollectionRequest describeCollectionRequest =
 DescribeCollectionRequest.builder()
                    .collectionId(collectionName)
                    .build();

            DescribeCollectionResponse describeCollectionResponse = rekClient
                    .describeCollection(describeCollectionRequest);
            System.out.println("Collection Arn : " +
 describeCollectionResponse.collectionARN());
            System.out.println("Created : " +
 describeCollectionResponse.creationTimestamp().toString());

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeCollection中的。

`DetectFaces`

以下代码示例演示了如何使用 `DetectFaces`。

有关更多信息，请参阅检测图像中的人脸。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectFaces {
    public static void main(String[] args) {
        final String usage = """

            Usage:   <bucketName> <sourceImage>

            Where:
                bucketName = The name of the Amazon S3 bucket where the source image
 is stored.
                sourceImage - The name of the source image file in the Amazon S3
 bucket. (for example, pic1.png).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectFacesinImage(rekClient, bucketName, sourceImage);
        rekClient.close();
    }
```

```
   /**
    * Detects faces in an image stored in an Amazon S3 bucket using the Amazon
Rekognition service.
    *
    * @param rekClient    The Amazon Rekognition client used to interact with the
Rekognition service.
    * @param bucketName   The name of the Amazon S3 bucket where the source image
is stored.
    * @param sourceImage  The name of the source image file in the Amazon S3
bucket.
    */
   public static void detectFacesinImage(RekognitionClient rekClient, String
bucketName, String sourceImage) {
       try {
           S3Object s3ObjectTarget = S3Object.builder()
               .bucket(bucketName)
               .name(sourceImage)
               .build();

           Image targetImage = Image.builder()
               .s3Object(s3ObjectTarget)
               .build();

           DetectFacesRequest facesRequest = DetectFacesRequest.builder()
               .attributes(Attribute.ALL)
               .image(targetImage)
               .build();

           DetectFacesResponse facesResponse = rekClient.detectFaces(facesRequest);
           List<FaceDetail> faceDetails = facesResponse.faceDetails();
           for (FaceDetail face : faceDetails) {
               AgeRange ageRange = face.ageRange();
               System.out.println("The detected face is estimated to be between "
                       + ageRange.low().toString() + " and " +
ageRange.high().toString()
                       + " years old.");

               System.out.println("There is a smile : " +
face.smile().value().toString());
           }

       } catch (RekognitionException e) {
           System.out.println(e.getMessage());
```

```
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DetectFaces中的。

`DetectLabels`

以下代码示例演示了如何使用 `DetectLabels`。

有关更多信息，请参阅检测图像中的标签。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectLabels {
    public static void main(String[] args) {
```

```
        final String usage = """
            Usage: <bucketName> <sourceImage>

            Where:
                bucketName - The name of the Amazon S3 bucket where the image is
stored
                sourceImage - The name of the image file (for example, pic1.png).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0] ;
        String sourceImage = args[1] ;
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectImageLabels(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Detects the labels in an image stored in an Amazon S3 bucket using the Amazon
Rekognition service.
     *
     * @param rekClient     the Amazon Rekognition client used to make the detection
request
     * @param bucketName    the name of the Amazon S3 bucket where the image is
stored
     * @param sourceImage   the name of the image file to be analyzed
     */
    public static void detectImageLabels(RekognitionClient rekClient, String
bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

            Image souImage = Image.builder()
```

```
                    .s3Object(s3ObjectTarget)
                    .build();

            DetectLabelsRequest detectLabelsRequest = DetectLabelsRequest.builder()
                    .image(souImage)
                    .maxLabels(10)
                    .build();

            DetectLabelsResponse labelsResponse =
 rekClient.detectLabels(detectLabelsRequest);
            List<Label> labels = labelsResponse.labels();
            System.out.println("Detected labels for the given photo");
            for (Label label : labels) {
                System.out.println(label.name() + ": " +
 label.confidence().toString());
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DetectLabels中的。

`DetectModerationLabels`

以下代码示例演示了如何使用 `DetectModerationLabels`。

有关更多信息，请参阅检测不适宜的图像。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectModerationLabels {

    public static void main(String[] args) {
        final String usage = """
            Usage:  <bucketName>  <sourceImage>

            Where:
                bucketName - The name of the S3 bucket where the images are stored.
                sourceImage - The name of the image (for example, pic1.png).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectModLabels(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
```

```
    * Detects moderation labels in an image stored in an Amazon S3 bucket.
    *
    * @param rekClient      the Amazon Rekognition client to use for the detection
    * @param bucketName     the name of the Amazon S3 bucket where the image is
stored
    * @param sourceImage    the name of the image file to be analyzed
    *
    * @throws RekognitionException if there is an error during the image detection
process
    */
   public static void detectModLabels(RekognitionClient rekClient, String
bucketName, String sourceImage) {
       try {
           S3Object s3ObjectTarget = S3Object.builder()
                   .bucket(bucketName)
                   .name(sourceImage)
                   .build();

           Image targetImage = Image.builder()
                   .s3Object(s3ObjectTarget)
                   .build();

           DetectModerationLabelsRequest moderationLabelsRequest =
DetectModerationLabelsRequest.builder()
                   .image(targetImage)
                   .minConfidence(60F)
                   .build();

           DetectModerationLabelsResponse moderationLabelsResponse = rekClient
                   .detectModerationLabels(moderationLabelsRequest);
           List<ModerationLabel> labels =
moderationLabelsResponse.moderationLabels();
           System.out.println("Detected labels for image");
           for (ModerationLabel label : labels) {
               System.out.println("Label: " + label.name()
                       + "\n Confidence: " + label.confidence().toString() + "%"
                       + "\n Parent:" + label.parentName());
           }

       } catch (RekognitionException e) {
           e.printStackTrace();
           System.exit(1);
       }
   }
```

```
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DetectModerationLabels中的。

`DetectText`

以下代码示例演示了如何使用 `DetectText`。

有关更多信息，请参阅检测图像中的文本。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.*;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class DetectText {
    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:   <bucketName> <sourceImage>\n" +
            "\n" +
```

```
            "Where:\n" +
            "   bucketName - The name of the S3 bucket where the image is stored\n"
+
            "   sourceImage - The path to the image that contains text (for example,
pic1.png). \n";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String sourceImage = args[1];
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        detectTextLabels(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Detects text labels in an image stored in an S3 bucket using Amazon
Rekognition.
     *
     * @param rekClient    an instance of the Amazon Rekognition client
     * @param bucketName   the name of the S3 bucket where the image is stored
     * @param sourceImage  the name of the image file in the S3 bucket
     * @throws RekognitionException if an error occurs while calling the Amazon
Rekognition API
     */
    public static void detectTextLabels(RekognitionClient rekClient, String
bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                    .bucket(bucketName)
                    .name(sourceImage)
                    .build();

            Image souImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();
```

```
            DetectTextRequest textRequest = DetectTextRequest.builder()
                    .image(souImage)
                    .build();

            DetectTextResponse textResponse = rekClient.detectText(textRequest);
            List<TextDetection> textCollection = textResponse.textDetections();
            System.out.println("Detected lines and words");
            for (TextDetection text : textCollection) {
                System.out.println("Detected: " + text.detectedText());
                System.out.println("Confidence: " + text.confidence().toString());
                System.out.println("Id : " + text.id());
                System.out.println("Parent Id: " + text.parentId());
                System.out.println("Type: " + text.type());
                System.out.println();
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DetectText中的。

`IndexFaces`

以下代码示例演示了如何使用 `IndexFaces`。

有关更多信息，请参阅将人脸添加到集合中。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
```

```
import software.amazon.awssdk.services.rekognition.model.*;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AddFacesToCollection {
    public static void main(String[] args) {
        final String usage = """
            Usage: <collectionId> <sourceImage> <bucketName>

            Where:
                collectionName - The name of the collection.
                sourceImage - The name of the image (for example, pic1.png).
                bucketName - The name of the S3 bucket.
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String sourceImage = args[1];
        String bucketName = args[2];;
        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        addToCollection(rekClient, collectionId, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Adds a face from an image to an Amazon Rekognition collection.
     *
     * @param rekClient     the Amazon Rekognition client
     * @param collectionId  the ID of the collection to add the face to
```

```
    * @param bucketName    the name of the Amazon S3 bucket containing the image
    * @param sourceImage   the name of the image file to add to the collection
    * @throws RekognitionException if there is an error while interacting with the
Amazon Rekognition service
    */
   public static void addToCollection(RekognitionClient rekClient, String
collectionId, String bucketName, String sourceImage) {
       try {
           S3Object s3ObjectTarget = S3Object.builder()
                   .bucket(bucketName)
                   .name(sourceImage)
                   .build();

           Image targetImage = Image.builder()
                   .s3Object(s3ObjectTarget)
                   .build();

           IndexFacesRequest facesRequest = IndexFacesRequest.builder()
                   .collectionId(collectionId)
                   .image(targetImage)
                   .maxFaces(1)
                   .qualityFilter(QualityFilter.AUTO)
                   .detectionAttributes(Attribute.DEFAULT)
                   .build();

           IndexFacesResponse facesResponse = rekClient.indexFaces(facesRequest);
           System.out.println("Results for the image");
           System.out.println("\n Faces indexed:");
           List<FaceRecord> faceRecords = facesResponse.faceRecords();
           for (FaceRecord faceRecord : faceRecords) {
               System.out.println("  Face ID: " + faceRecord.face().faceId());
               System.out.println("  Location:" +
faceRecord.faceDetail().boundingBox().toString());
           }

           List<UnindexedFace> unindexedFaces = facesResponse.unindexedFaces();
           System.out.println("Faces not indexed:");
           for (UnindexedFace unindexedFace : unindexedFaces) {
               System.out.println("  Location:" +
unindexedFace.faceDetail().boundingBox().toString());
               System.out.println("  Reasons:");
               for (Reason reason : unindexedFace.reasons()) {
                   System.out.println("Reason:  " + reason);
               }
```

```
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考IndexFaces中的。

`ListCollections`

以下代码示例演示了如何使用 `ListCollections`。

有关更多信息，请参阅列出集合。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListCollections中的。

`ListFaces`

以下代码示例演示了如何使用 `ListFaces`。

有关更多信息，请参阅列出集合中的人脸。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListFaces中的。

`RecognizeCelebrities`

以下代码示例演示了如何使用 `RecognizeCelebrities`。

有关更多信息，请参阅识别图像中的名人。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.core.SdkBytes;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

import software.amazon.awssdk.services.rekognition.model.*;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RecognizeCelebrities {
    public static void main(String[] args) {
        final String usage = """
                Usage:   <bucketName> <sourceImage>

                Where:
                   bucketName - The name of the S3 bucket where the images are
 stored.
                   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
       }
```

```
        String bucketName = args[0];;
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Locating celebrities in " + sourceImage);
        recognizeAllCelebrities(rekClient, bucketName, sourceImage);
        rekClient.close();
    }

    /**
     * Recognizes all celebrities in an image stored in an Amazon S3 bucket.
     *
     * @param rekClient    the Amazon Rekognition client used to perform the
celebrity recognition operation
     * @param bucketName   the name of the Amazon S3 bucket where the source image
is stored
     * @param sourceImage  the name of the source image file stored in the Amazon S3
bucket
     */
    public static void recognizeAllCelebrities(RekognitionClient rekClient, String
bucketName, String sourceImage) {
        try {
            S3Object s3ObjectTarget = S3Object.builder()
                .bucket(bucketName)
                .name(sourceImage)
                .build();

            Image souImage = Image.builder()
                    .s3Object(s3ObjectTarget)
                    .build();

            RecognizeCelebritiesRequest request =
RecognizeCelebritiesRequest.builder()
                    .image(souImage)
                    .build();

            RecognizeCelebritiesResponse result =
rekClient.recognizeCelebrities(request);
            List<Celebrity> celebs = result.celebrityFaces();
            System.out.println(celebs.size() + " celebrity(s) were recognized.\n");
```

```
            for (Celebrity celebrity : celebs) {
                System.out.println("Celebrity recognized: " + celebrity.name());
                System.out.println("Celebrity ID: " + celebrity.id());

                System.out.println("Further information (if available):");
                for (String url : celebrity.urls()) {
                    System.out.println(url);
                }
                System.out.println();
            }
            System.out.println(result.unrecognizedFaces().size() + " face(s) were
 unrecognized.");

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[RecognizeCelebrities](#)中的。

`SearchFaces`

以下代码示例演示了如何使用 `SearchFaces`。

有关更多信息，请参阅[搜索人脸（人脸 ID）](#)。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.SearchFacesByImageRequest;
```

```
import software.amazon.awssdk.services.rekognition.model.Image;
import software.amazon.awssdk.services.rekognition.model.SearchFacesByImageResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SearchFaceMatchingImageCollection {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <collectionId> <sourceImage>

                Where:
                   collectionId - The id of the collection. \s
                   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png).\s

                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String collectionId = args[0];
        String sourceImage = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Searching for a face in a collections");
        searchFaceInCollection(rekClient, collectionId, sourceImage);
```

```
        rekClient.close();
    }

    public static void searchFaceInCollection(RekognitionClient rekClient, String
 collectionId, String sourceImage) {
        try {
            InputStream sourceStream = new FileInputStream(new File(sourceImage));
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
            Image souImage = Image.builder()
                    .bytes(sourceBytes)
                    .build();

            SearchFacesByImageRequest facesByImageRequest =
 SearchFacesByImageRequest.builder()
                    .image(souImage)
                    .maxFaces(10)
                    .faceMatchThreshold(70F)
                    .collectionId(collectionId)
                    .build();

            SearchFacesByImageResponse imageResponse =
 rekClient.searchFacesByImage(facesByImageRequest);
            System.out.println("Faces matching in the collection");
            List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
            for (FaceMatch face : faceImageMatches) {
                System.out.println("The similarity level is  " + face.similarity());
                System.out.println();
            }

        } catch (RekognitionException | FileNotFoundException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SearchFaces中的。

SearchFacesByImage

以下代码示例演示了如何使用 SearchFacesByImage。

有关更多信息，请参阅搜索人脸（图像）。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.SearchFacesRequest;
import software.amazon.awssdk.services.rekognition.model.SearchFacesResponse;
import software.amazon.awssdk.services.rekognition.model.FaceMatch;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SearchFaceMatchingIdCollection {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <collectionId> <sourceImage>

                Where:
                   collectionId - The id of the collection. \s
                   sourceImage - The path to the image (for example, C:\\AWS\
\pic1.png).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String collectionId = args[0];
        String faceId = args[1];
        Region region = Region.US_WEST_2;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        System.out.println("Searching for a face in a collections");
        searchFacebyId(rekClient, collectionId, faceId);
        rekClient.close();
    }

    public static void searchFacebyId(RekognitionClient rekClient, String
 collectionId, String faceId) {
        try {
            SearchFacesRequest searchFacesRequest = SearchFacesRequest.builder()
                    .collectionId(collectionId)
                    .faceId(faceId)
                    .faceMatchThreshold(70F)
                    .maxFaces(2)
                    .build();

            SearchFacesResponse imageResponse =
 rekClient.searchFaces(searchFacesRequest);
            System.out.println("Faces matching in the collection");
            List<FaceMatch> faceImageMatches = imageResponse.faceMatches();
            for (FaceMatch face : faceImageMatches) {
                System.out.println("The similarity level is  " + face.similarity());
                System.out.println();
            }

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SearchFacesByImage中的。

场景

创建无服务器应用程序来管理照片

以下代码示例演示如何创建无服务器应用程序，让用户能够使用标签管理照片。

适用于 Java 的 SDK 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 Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

检测图像中的 PPE

以下代码示例展示如何构建采用 Amazon Rekognition 来检测图像中的个人防护设备（PPE）的应用程序。

适用于 Java 的 SDK 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
- Amazon SES

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。

适用于 Java 的 SDK 2.x

> **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。

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 vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartCelebrityRecognitionRequest recognitionRequest =
StartCelebrityRecognitionRequest.builder()
                    .jobTag("Celebrities")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .build();

            StartCelebrityRecognitionResponse startCelebrityRecognitionResult =
rekClient
                    .startCelebrityRecognition(recognitionRequest);
            startJobId = startCelebrityRecognitionResult.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getCelebrityDetectionResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
```

```
            GetCelebrityRecognitionResponse recognitionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (recognitionResponse != null)
                    paginationToken = recognitionResponse.nextToken();

                GetCelebrityRecognitionRequest recognitionRequest =
GetCelebrityRecognitionRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .sortBy(CelebrityRecognitionSortBy.TIMESTAMP)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds
                while (!finished) {
                    recognitionResponse =
rekClient.getCelebrityRecognition(recognitionRequest);
                    status = recognitionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is null.
                VideoMetadata videoMetaData = recognitionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<CelebrityRecognition> celebs =
recognitionResponse.celebrities();
```

```
                for (CelebrityRecognition celeb : celebs) {
                    long seconds = celeb.timestamp() / 1000;
                    System.out.print("Sec: " + seconds + " ");
                    CelebrityDetail details = celeb.celebrity();
                    System.out.println("Name: " + details.name());
                    System.out.println("Id: " + details.id());
                    System.out.println();
                }

            } while (recognitionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

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 vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .minConfidence(50F)
                    .build();

            StartLabelDetectionResponse labelDetectionResponse =
rekClient.startLabelDetection(labelDetectionRequest);
            startJobId = labelDetectionResponse.jobId();
```

```
            boolean ans = true;
            String status = "";
            int yy = 0;
            while (ans) {

                GetLabelDetectionRequest detectionRequest =
GetLabelDetectionRequest.builder()
                        .jobId(startJobId)
                        .maxResults(10)
                        .build();

                GetLabelDetectionResponse result =
rekClient.getLabelDetection(detectionRequest);
                status = result.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    ans = false;
                else
                    System.out.println(yy + " status is: " + status);

                Thread.sleep(1000);
                yy++;
            }

            System.out.println(startJobId + " status is: " + status);

        } catch (RekognitionException | InterruptedException e) {
            e.getMessage();
            System.exit(1);
        }
    }

    public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {
        List<Message> messages;
        ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
                .queueUrl(queueUrl)
                .build();

        try {
            messages = sqs.receiveMessage(messageRequest).messages();

            if (!messages.isEmpty()) {
```

```
                for (Message message : messages) {
                    String notification = message.body();

                    // Get the status and job id from the notification
                    ObjectMapper mapper = new ObjectMapper();
                    JsonNode jsonMessageTree = mapper.readTree(notification);
                    JsonNode messageBodyText = jsonMessageTree.get("Message");
                    ObjectMapper operationResultMapper = new ObjectMapper();
                    JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                    JsonNode operationJobId = jsonResultTree.get("JobId");
                    JsonNode operationStatus = jsonResultTree.get("Status");
                    System.out.println("Job found in JSON is " + operationJobId);

                    DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                            .queueUrl(queueUrl)
                            .build();

                    String jobId = operationJobId.textValue();
                    if (startJobId.compareTo(jobId) == 0) {
                        System.out.println("Job id: " + operationJobId);
                        System.out.println("Status : " +
operationStatus.toString());

                        if (operationStatus.asText().equals("SUCCEEDED"))
                            getResultsLabels(rekClient);
                        else
                            System.out.println("Video analysis failed");

                        sqs.deleteMessage(deleteMessageRequest);
                    } else {
                        System.out.println("Job received was not job " +
startJobId);
                        sqs.deleteMessage(deleteMessageRequest);
                    }
                }
            }

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        } catch (JsonMappingException e) {
            e.printStackTrace();
```

```
        } catch (JsonProcessingException e) {
            e.printStackTrace();
        }
    }

    // Gets the job results by calling GetLabelDetection
    private static void getResultsLabels(RekognitionClient rekClient) {

        int maxResults = 10;
        String paginationToken = null;
        GetLabelDetectionResponse labelDetectionResult = null;

        try {
            do {
                if (labelDetectionResult != null)
                    paginationToken = labelDetectionResult.nextToken();

                GetLabelDetectionRequest labelDetectionRequest =
GetLabelDetectionRequest.builder()
                        .jobId(startJobId)
                        .sortBy(LabelDetectionSortBy.TIMESTAMP)
                        .maxResults(maxResults)
                        .nextToken(paginationToken)
                        .build();

                labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
                VideoMetadata videoMetaData = labelDetectionResult.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());

                List<LabelDetection> detectedLabels = labelDetectionResult.labels();
                for (LabelDetection detectedLabel : detectedLabels) {
                    long seconds = detectedLabel.timestamp();
                    Label label = detectedLabel.label();
                    System.out.println("Millisecond: " + seconds + " ");

                    System.out.println("   Label:" + label.name());
                    System.out.println("   Confidence:" +
detectedLabel.label().confidence().toString());

                    List<Instance> instances = label.instances();
```

```
                    System.out.println("   Instances of " + label.name());

                    if (instances.isEmpty()) {
                        System.out.println("        " + "None");
                    } else {
                        for (Instance instance : instances) {
                            System.out.println("        Confidence: " +
 instance.confidence().toString());
                            System.out.println("        Bounding box: " +
 instance.boundingBox().toString());
                        }
                    }
                    System.out.println("   Parent labels for " + label.name() +
 ":");
                    List<Parent> parents = label.parents();

                    if (parents.isEmpty()) {
                        System.out.println("        None");
                    } else {
                        for (Parent parent : parents) {
                            System.out.println("   " + parent.name());
                        }
                    }
                    System.out.println();
                }
            } while (labelDetectionResult != null &&
 labelDetectionResult.nextToken() != null);

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        }
    }
}
```

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 vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartLabelDetectionRequest labelDetectionRequest =
StartLabelDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .minConfidence(50F)
                    .build();

            StartLabelDetectionResponse labelDetectionResponse =
rekClient.startLabelDetection(labelDetectionRequest);
            startJobId = labelDetectionResponse.jobId();

            boolean ans = true;
            String status = "";
            int yy = 0;
            while (ans) {

                GetLabelDetectionRequest detectionRequest =
GetLabelDetectionRequest.builder()
                        .jobId(startJobId)
                        .maxResults(10)
                        .build();

                GetLabelDetectionResponse result =
rekClient.getLabelDetection(detectionRequest);
                status = result.jobStatusAsString();

                if (status.compareTo("SUCCEEDED") == 0)
                    ans = false;
                else
                    System.out.println(yy + " status is: " + status);

                Thread.sleep(1000);
                yy++;
            }

            System.out.println(startJobId + " status is: " + status);
```

```
        } catch (RekognitionException | InterruptedException e) {
            e.getMessage();
            System.exit(1);
        }
    }

    public static void getLabelJob(RekognitionClient rekClient, SqsClient sqs,
String queueUrl) {
        List<Message> messages;
        ReceiveMessageRequest messageRequest = ReceiveMessageRequest.builder()
                .queueUrl(queueUrl)
                .build();

        try {
            messages = sqs.receiveMessage(messageRequest).messages();

            if (!messages.isEmpty()) {
                for (Message message : messages) {
                    String notification = message.body();

                    // Get the status and job id from the notification
                    ObjectMapper mapper = new ObjectMapper();
                    JsonNode jsonMessageTree = mapper.readTree(notification);
                    JsonNode messageBodyText = jsonMessageTree.get("Message");
                    ObjectMapper operationResultMapper = new ObjectMapper();
                    JsonNode jsonResultTree =
operationResultMapper.readTree(messageBodyText.textValue());
                    JsonNode operationJobId = jsonResultTree.get("JobId");
                    JsonNode operationStatus = jsonResultTree.get("Status");
                    System.out.println("Job found in JSON is " + operationJobId);

                    DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                            .queueUrl(queueUrl)
                            .build();

                    String jobId = operationJobId.textValue();
                    if (startJobId.compareTo(jobId) == 0) {
                        System.out.println("Job id: " + operationJobId);
                        System.out.println("Status : " +
operationStatus.toString());

                        if (operationStatus.asText().equals("SUCCEEDED"))
```

```
                            getResultsLabels(rekClient);
                        else
                            System.out.println("Video analysis failed");

                        sqs.deleteMessage(deleteMessageRequest);
                    } else {
                        System.out.println("Job received was not job " +
startJobId);
                        sqs.deleteMessage(deleteMessageRequest);
                    }
                }
            }

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        } catch (JsonMappingException e) {
            e.printStackTrace();
        } catch (JsonProcessingException e) {
            e.printStackTrace();
        }
    }

    // Gets the job results by calling GetLabelDetection
    private static void getResultsLabels(RekognitionClient rekClient) {

        int maxResults = 10;
        String paginationToken = null;
        GetLabelDetectionResponse labelDetectionResult = null;

        try {
            do {
                if (labelDetectionResult != null)
                    paginationToken = labelDetectionResult.nextToken();

                GetLabelDetectionRequest labelDetectionRequest =
GetLabelDetectionRequest.builder()
                        .jobId(startJobId)
                        .sortBy(LabelDetectionSortBy.TIMESTAMP)
                        .maxResults(maxResults)
                        .nextToken(paginationToken)
                        .build();
```

```
                labelDetectionResult =
rekClient.getLabelDetection(labelDetectionRequest);
                VideoMetadata videoMetaData = labelDetectionResult.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());

                List<LabelDetection> detectedLabels = labelDetectionResult.labels();
                for (LabelDetection detectedLabel : detectedLabels) {
                    long seconds = detectedLabel.timestamp();
                    Label label = detectedLabel.label();
                    System.out.println("Millisecond: " + seconds + " ");

                    System.out.println("   Label:" + label.name());
                    System.out.println("   Confidence:" +
detectedLabel.label().confidence().toString());

                    List<Instance> instances = label.instances();
                    System.out.println("   Instances of " + label.name());

                    if (instances.isEmpty()) {
                        System.out.println("        " + "None");
                    } else {
                        for (Instance instance : instances) {
                            System.out.println("        Confidence: " +
instance.confidence().toString());
                            System.out.println("        Bounding box: " +
instance.boundingBox().toString());
                        }
                    }
                    System.out.println("   Parent labels for " + label.name() +
":");
                    List<Parent> parents = label.parents();

                    if (parents.isEmpty()) {
                        System.out.println("        None");
                    } else {
                        for (Parent parent : parents) {
                            System.out.println("   " + parent.name());
                        }
                    }
                    System.out.println();
                }
```

```
            } while (labelDetectionResult != null &&
 labelDetectionResult.nextToken() != null);

        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        }
    }
}
```

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 vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartContentModerationRequest modDetectionRequest =
StartContentModerationRequest.builder()
                    .jobTag("Moderation")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .build();

            StartContentModerationResponse startModDetectionResult = rekClient
                    .startContentModeration(modDetectionRequest);
            startJobId = startModDetectionResult.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getModResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetContentModerationResponse modDetectionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (modDetectionResponse != null)
                    paginationToken = modDetectionResponse.nextToken();

                GetContentModerationRequest modRequest =
GetContentModerationRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
```

```
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds.
                while (!finished) {
                    modDetectionResponse =
rekClient.getContentModeration(modRequest);
                    status = modDetectionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is null.
                VideoMetadata videoMetaData = modDetectionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<ContentModerationDetection> mods =
modDetectionResponse.moderationLabels();
                for (ContentModerationDetection mod : mods) {
                    long seconds = mod.timestamp() / 1000;
                    System.out.print("Mod label: " + seconds + " ");
                    System.out.println(mod.moderationLabel().toString());
                    System.out.println();
                }

            } while (modDetectionResponse != null &&
modDetectionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
```

```
    }
}
```

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(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();

            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartShotDetectionFilter cueDetectionFilter =
StartShotDetectionFilter.builder()
                    .minSegmentConfidence(60F)
                    .build();

            StartTechnicalCueDetectionFilter technicalCueDetectionFilter =
StartTechnicalCueDetectionFilter.builder()
                    .minSegmentConfidence(60F)
                    .build();

            StartSegmentDetectionFilters filters =
StartSegmentDetectionFilters.builder()
                    .shotFilter(cueDetectionFilter)
                    .technicalCueFilter(technicalCueDetectionFilter)
                    .build();

            StartSegmentDetectionRequest segDetectionRequest =
StartSegmentDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .segmentTypes(SegmentType.TECHNICAL_CUE, SegmentType.SHOT)
                    .video(vidOb)
                    .filters(filters)
                    .build();

            StartSegmentDetectionResponse segDetectionResponse =
rekClient.startSegmentDetection(segDetectionRequest);
            startJobId = segDetectionResponse.jobId();
```

```
        } catch (RekognitionException e) {
            e.getMessage();
            System.exit(1);
        }
    }

    public static void getSegmentResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetSegmentDetectionResponse segDetectionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (segDetectionResponse != null)
                    paginationToken = segDetectionResponse.nextToken();

                GetSegmentDetectionRequest recognitionRequest =
GetSegmentDetectionRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds.
                while (!finished) {
                    segDetectionResponse =
rekClient.getSegmentDetection(recognitionRequest);
                    status = segDetectionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }
                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is null.
```

```
                List<VideoMetadata> videoMetaData =
segDetectionResponse.videoMetadata();
                for (VideoMetadata metaData : videoMetaData) {
                    System.out.println("Format: " + metaData.format());
                    System.out.println("Codec: " + metaData.codec());
                    System.out.println("Duration: " + metaData.durationMillis());
                    System.out.println("FrameRate: " + metaData.frameRate());
                    System.out.println("Job");
                }

                List<SegmentDetection> detectedSegments =
segDetectionResponse.segments();
                for (SegmentDetection detectedSegment : detectedSegments) {
                    String type = detectedSegment.type().toString();
                    if (type.contains(SegmentType.TECHNICAL_CUE.toString())) {
                        System.out.println("Technical Cue");
                        TechnicalCueSegment segmentCue =
detectedSegment.technicalCueSegment();
                        System.out.println("\tType: " + segmentCue.type());
                        System.out.println("\tConfidence: " +
segmentCue.confidence().toString());
                    }

                    if (type.contains(SegmentType.SHOT.toString())) {
                        System.out.println("Shot");
                        ShotSegment segmentShot = detectedSegment.shotSegment();
                        System.out.println("\tIndex " + segmentShot.index());
                        System.out.println("\tConfidence: " +
segmentShot.confidence().toString());
                    }

                    long seconds = detectedSegment.durationMillis();
                    System.out.println("\tDuration : " + seconds + " milliseconds");
                    System.out.println("\tStart time code: " +
detectedSegment.startTimecodeSMPTE());
                    System.out.println("\tEnd time code: " +
detectedSegment.endTimecodeSMPTE());
                    System.out.println("\tDuration time code: " +
detectedSegment.durationSMPTE());
                    System.out.println();
                }

            } while (segDetectionResponse != null &&
segDetectionResponse.nextToken() != null);
```

```
        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

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.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.GetTextDetectionResponse;
import software.amazon.awssdk.services.rekognition.model.GetTextDetectionRequest;
import software.amazon.awssdk.services.rekognition.model.VideoMetadata;
import software.amazon.awssdk.services.rekognition.model.TextDetectionResult;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class VideoDetectText {
    private static String startJobId = "";

    public static void main(String[] args) {
        final String usage = """

                Usage:    <bucket> <video> <topicArn> <roleArn>

                Where:
```

```
                    bucket - The name of the bucket in which the video is located
(for example, (for example, myBucket).\s
                    video - The name of video (for example, people.mp4).\s
                    topicArn - The ARN of the Amazon Simple Notification Service
(Amazon SNS) topic.\s
                    roleArn - The ARN of the AWS Identity and Access Management (IAM)
role to use.\s
                """;

        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucket = args[0];
        String video = args[1];
        String topicArn = args[2];
        String roleArn = args[3];

        Region region = Region.US_EAST_1;
        RekognitionClient rekClient = RekognitionClient.builder()
                .region(region)
                .build();

        NotificationChannel channel = NotificationChannel.builder()
                .snsTopicArn(topicArn)
                .roleArn(roleArn)
                .build();

        startTextLabels(rekClient, channel, bucket, video);
        getTextResults(rekClient);
        System.out.println("This example is done!");
        rekClient.close();
    }

    public static void startTextLabels(RekognitionClient rekClient,
            NotificationChannel channel,
            String bucket,
            String video) {
        try {
            S3Object s3Obj = S3Object.builder()
                    .bucket(bucket)
                    .name(video)
                    .build();
```

```
            Video vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();

            StartTextDetectionRequest labelDetectionRequest =
StartTextDetectionRequest.builder()
                    .jobTag("DetectingLabels")
                    .notificationChannel(channel)
                    .video(vidOb)
                    .build();

            StartTextDetectionResponse labelDetectionResponse =
rekClient.startTextDetection(labelDetectionRequest);
            startJobId = labelDetectionResponse.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getTextResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetTextDetectionResponse textDetectionResponse = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (textDetectionResponse != null)
                    paginationToken = textDetectionResponse.nextToken();

                GetTextDetectionRequest recognitionRequest =
GetTextDetectionRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds.
                while (!finished) {
```

```
                    textDetectionResponse =
rekClient.getTextDetection(recognitionRequest);
                    status = textDetectionResponse.jobStatusAsString();

                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is null.
                VideoMetadata videoMetaData = textDetectionResponse.videoMetadata();
                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<TextDetectionResult> labels =
textDetectionResponse.textDetections();
                for (TextDetectionResult detectedText : labels) {
                    System.out.println("Confidence: " +
detectedText.textDetection().confidence().toString());
                    System.out.println("Id : " + detectedText.textDetection().id());
                    System.out.println("Parent Id: " +
detectedText.textDetection().parentId());
                    System.out.println("Type: " +
detectedText.textDetection().type());
                    System.out.println("Text: " +
detectedText.textDetection().detectedText());
                    System.out.println();
                }

            } while (textDetectionResponse != null &&
textDetectionResponse.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
```

```
        }
    }
}
```

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 vidOb = Video.builder()
                    .s3Object(s3Obj)
                    .build();
```

```
            StartPersonTrackingRequest personTrackingRequest =
StartPersonTrackingRequest.builder()
                    .jobTag("DetectingLabels")
                    .video(vidOb)
                    .notificationChannel(channel)
                    .build();

            StartPersonTrackingResponse labelDetectionResponse =
rekClient.startPersonTracking(personTrackingRequest);
            startJobId = labelDetectionResponse.jobId();

        } catch (RekognitionException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getPersonDetectionResults(RekognitionClient rekClient) {
        try {
            String paginationToken = null;
            GetPersonTrackingResponse personTrackingResult = null;
            boolean finished = false;
            String status;
            int yy = 0;

            do {
                if (personTrackingResult != null)
                    paginationToken = personTrackingResult.nextToken();

                GetPersonTrackingRequest recognitionRequest =
GetPersonTrackingRequest.builder()
                        .jobId(startJobId)
                        .nextToken(paginationToken)
                        .maxResults(10)
                        .build();

                // Wait until the job succeeds
                while (!finished) {

                    personTrackingResult =
rekClient.getPersonTracking(recognitionRequest);
                    status = personTrackingResult.jobStatusAsString();
```

```
                    if (status.compareTo("SUCCEEDED") == 0)
                        finished = true;
                    else {
                        System.out.println(yy + " status is: " + status);
                        Thread.sleep(1000);
                    }
                    yy++;
                }

                finished = false;

                // Proceed when the job is done - otherwise VideoMetadata is null.
                VideoMetadata videoMetaData = personTrackingResult.videoMetadata();

                System.out.println("Format: " + videoMetaData.format());
                System.out.println("Codec: " + videoMetaData.codec());
                System.out.println("Duration: " + videoMetaData.durationMillis());
                System.out.println("FrameRate: " + videoMetaData.frameRate());
                System.out.println("Job");

                List<PersonDetection> detectedPersons =
 personTrackingResult.persons();
                for (PersonDetection detectedPerson : detectedPersons) {
                    long seconds = detectedPerson.timestamp() / 1000;
                    System.out.print("Sec: " + seconds + " ");
                    System.out.println("Person Identifier: " +
 detectedPerson.person().index());
                    System.out.println();
                }

            } while (personTrackingResult != null &&
 personTrackingResult.nextToken() != null);

        } catch (RekognitionException | InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - GetCelebrityRecognition

- GetContentModeration
- GetLabelDetection
- GetPersonTracking
- GetSegmentDetection
- GetTextDetection
- StartCelebrityRecognition
- StartContentModeration
- StartLabelDetection
- StartPersonTracking
- StartSegmentDetection
- StartTextDetection

检测图像中的对象

以下代码示例演示如何构建采用 Amazon Rekognition 来按类别检测图像中物体的应用程序。

适用于 Java 的 SDK 2.x

展示如何使用 Amazon Rekognition Java API 创建应用程序，该应用程序采用 Amazon Rekognition 来按类别识别位于 Amazon Simple Storage Service (Amazon S3) 存储桶的图像当中的对象。该应用程序使用 Amazon Simple Email Service (Amazon SES) 向管理员发送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
- Amazon SES

检测视频中的人物和对象

以下代码示例演示如何使用 Amazon Rekognition 检测视频中的人物和物体。

适用于 Java 的 SDK 2.x

展示如何使用 Amazon Rekognition Java API 创建应用程序，以检测位于 Amazon Simple Storage Service (Amazon S3) 存储桶的视频当中的人脸和对象。该应用程序使用 Amazon Simple Email Service (Amazon SES) 向管理员发送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
- Amazon SES
- Amazon SNS
- Amazon SQS

使用 SDK for Java 2.x 的 Route 53 域注册示例

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。

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 基本功能
- 操作

开始使用

开始使用 Route 53 域注册

以下代码示例显示了如何开始使用 Route 53 域名注册。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListPrices中的。

基本功能

了解基本功能

以下代码示例展示了如何：

- 列出当前域，并列出过去一年的操作。
- 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。

适用于 Java 的 SDK 2.x

> **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。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This example uses pagination methods where applicable. For example, to list
 * domains, the
 * listDomainsPaginator method is used. For more information about pagination,
 * see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/pagination.html
 *
 * This Java code example performs the following operations:
 *
 * 1. List current domains.
 * 2. List operations in the past year.
 * 3. View billing for the account in the past year.
 * 4. View prices for domain types.
 * 5. Get domain suggestions.
 * 6. Check domain availability.
 * 7. Check domain transferability.
 * 8. Request a domain registration.
 * 9. Get operation details.
 * 10. Optionally, get domain details.
 */

public class Route53Scenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <domainType> <phoneNumber> <email> <domainSuggestion>
 <firstName> <lastName> <city>

                Where:
                    domainType - The domain type (for example, com).\s
                    phoneNumber - The phone number to use (for example,
 +91.9966564xxx)      email - The email address to use.      domainSuggestion - The
 domain suggestion (for example, findmy.accountants).\s
                    firstName - The first name to use to register a domain.\s
                    lastName -  The last name to use to register a domain.\s
                    city - the city to use to register a domain.\s
                    """;
```

```
        if (args.length != 7) {
            System.out.println(usage);
            System.exit(1);
        }

        String domainType = args[0];
        String phoneNumber = args[1];
        String email = args[2];
        String domainSuggestion = args[3];
        String firstName = args[4];
        String lastName = args[5];
        String city = args[6];
        Region region = Region.US_EAST_1;
        Route53DomainsClient route53DomainsClient = Route53DomainsClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Route 53 domains example
scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. List current domains.");
        listDomains(route53DomainsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. List operations in the past year.");
        listOperations(route53DomainsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. View billing for the account in the past year.");
        listBillingRecords(route53DomainsClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. View prices for domain types.");
        listPrices(route53DomainsClient, domainType);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("5. Get domain suggestions.");
        listDomainSuggestions(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Check domain availability.");
        checkDomainAvailability(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Check domain transferability.");
        checkDomainTransferability(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Request a domain registration.");
        String opId = requestDomainRegistration(route53DomainsClient,
domainSuggestion, phoneNumber, email, firstName,
                lastName, city);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get operation details.");
        getOperationalDetail(route53DomainsClient, opId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get domain details.");
        System.out.println("Note: You must have a registered domain to get
details.");
        System.out.println("Otherwise, an exception is thrown that states ");
        System.out.println("Domain xxxxxxx not found in xxxxxxx account.");
        getDomainDetails(route53DomainsClient, domainSuggestion);
        System.out.println(DASHES);
    }

    public static void getDomainDetails(Route53DomainsClient route53DomainsClient,
String domainSuggestion) {
        try {
            GetDomainDetailRequest detailRequest = GetDomainDetailRequest.builder()
                    .domainName(domainSuggestion)
                    .build();
```

```
            GetDomainDetailResponse response =
route53DomainsClient.getDomainDetail(detailRequest);
            System.out.println("The contact first name is " +
response.registrantContact().firstName());
            System.out.println("The contact last name is " +
response.registrantContact().lastName());
            System.out.println("The contact org name is " +
response.registrantContact().organizationName());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void getOperationalDetail(Route53DomainsClient
route53DomainsClient, String operationId) {
        try {
            GetOperationDetailRequest detailRequest =
GetOperationDetailRequest.builder()
                    .operationId(operationId)
                    .build();

            GetOperationDetailResponse response =
route53DomainsClient.getOperationDetail(detailRequest);
            System.out.println("Operation detail message is " + response.message());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static String requestDomainRegistration(Route53DomainsClient
route53DomainsClient,
            String domainSuggestion,
            String phoneNumber,
            String email,
            String firstName,
            String lastName,
            String city) {

        try {
            ContactDetail contactDetail = ContactDetail.builder()
```

```
                    .contactType(ContactType.COMPANY)
                    .state("LA")
                    .countryCode(CountryCode.IN)
                    .email(email)
                    .firstName(firstName)
                    .lastName(lastName)
                    .city(city)
                    .phoneNumber(phoneNumber)
                    .organizationName("My Org")
                    .addressLine1("My Address")
                    .zipCode("123 123")
                    .build();

            RegisterDomainRequest domainRequest = RegisterDomainRequest.builder()
                    .adminContact(contactDetail)
                    .registrantContact(contactDetail)
                    .techContact(contactDetail)
                    .domainName(domainSuggestion)
                    .autoRenew(true)
                    .durationInYears(1)
                    .build();

            RegisterDomainResponse response =
route53DomainsClient.registerDomain(domainRequest);
            System.out.println("Registration requested. Operation Id: " +
response.operationId());
            return response.operationId();

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static void checkDomainTransferability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
        try {
            CheckDomainTransferabilityRequest transferabilityRequest =
CheckDomainTransferabilityRequest.builder()
                    .domainName(domainSuggestion)
                    .build();

            CheckDomainTransferabilityResponse response = route53DomainsClient
```

```
                    .checkDomainTransferability(transferabilityRequest);
            System.out.println("Transferability: " +
response.transferability().transferable().toString());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void checkDomainAvailability(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
        try {
            CheckDomainAvailabilityRequest availabilityRequest =
CheckDomainAvailabilityRequest.builder()
                    .domainName(domainSuggestion)
                    .build();

            CheckDomainAvailabilityResponse response = route53DomainsClient
                    .checkDomainAvailability(availabilityRequest);
            System.out.println(domainSuggestion + " is " +
response.availability().toString());

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listDomainSuggestions(Route53DomainsClient
route53DomainsClient, String domainSuggestion) {
        try {
            GetDomainSuggestionsRequest suggestionsRequest =
GetDomainSuggestionsRequest.builder()
                    .domainName(domainSuggestion)
                    .suggestionCount(5)
                    .onlyAvailable(true)
                    .build();

            GetDomainSuggestionsResponse response =
route53DomainsClient.getDomainSuggestions(suggestionsRequest);
            List<DomainSuggestion> suggestions = response.suggestionsList();
            for (DomainSuggestion suggestion : suggestions) {
                System.out.println("Suggestion Name: " + suggestion.domainName());
```

```
                System.out.println("Availability: " + suggestion.availability());
                System.out.println(" ");
            }

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listPrices(Route53DomainsClient route53DomainsClient, String
domainType) {
        try {
            ListPricesRequest pricesRequest = ListPricesRequest.builder()
                    .tld(domainType)
                    .build();

            ListPricesIterable listRes =
route53DomainsClient.listPricesPaginator(pricesRequest);
            listRes.stream()
                    .flatMap(r -> r.prices().stream())
                    .forEach(content -> System.out.println(" Name: " +
content.name() +
                            " Registration: " + content.registrationPrice().price()
+ " "
                            + content.registrationPrice().currency() +
                            " Renewal: " + content.renewalPrice().price() + " " +
content.renewalPrice().currency()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listBillingRecords(Route53DomainsClient route53DomainsClient)
{
        try {
            Date currentDate = new Date();
            LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
            ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
            LocalDateTime localDateTime2 = localDateTime.minusYears(1);
            Instant myStartTime = localDateTime2.toInstant(zoneOffset);
```

```
            Instant myEndTime = localDateTime.toInstant(zoneOffset);

            ViewBillingRequest viewBillingRequest = ViewBillingRequest.builder()
                    .start(myStartTime)
                    .end(myEndTime)
                    .build();

            ViewBillingIterable listRes =
route53DomainsClient.viewBillingPaginator(viewBillingRequest);
            listRes.stream()
                    .flatMap(r -> r.billingRecords().stream())
                    .forEach(content -> System.out.println(" Bill Date:: " +
content.billDate() +
                            " Operation: " + content.operationAsString() +
                            " Price: " + content.price()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listOperations(Route53DomainsClient route53DomainsClient) {
        try {
            Date currentDate = new Date();
            LocalDateTime localDateTime =
currentDate.toInstant().atZone(ZoneId.systemDefault()).toLocalDateTime();
            ZoneOffset zoneOffset = ZoneOffset.of("+01:00");
            localDateTime = localDateTime.minusYears(1);
            Instant myTime = localDateTime.toInstant(zoneOffset);

            ListOperationsRequest operationsRequest =
ListOperationsRequest.builder()
                    .submittedSince(myTime)
                    .build();

            ListOperationsIterable listRes =
route53DomainsClient.listOperationsPaginator(operationsRequest);
            listRes.stream()
                    .flatMap(r -> r.operations().stream())
                    .forEach(content -> System.out.println(" Operation Id: " +
content.operationId() +
                            " Status: " + content.statusAsString() +
                            " Date: " + content.submittedDate()));
```

```
        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void listDomains(Route53DomainsClient route53DomainsClient) {
        try {
            ListDomainsIterable listRes =
 route53DomainsClient.listDomainsPaginator();
            listRes.stream()
                    .flatMap(r -> r.domains().stream())
                    .forEach(content -> System.out.println("The domain name is " +
 content.domainName()));

        } catch (Route53Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - CheckDomainAvailability
  - CheckDomainTransferability
  - GetDomainDetail
  - GetDomainSuggestions
  - GetOperationDetail
  - ListDomains
  - ListOperations
  - ListPrices
  - RegisterDomain
  - ViewBilling

操作

CheckDomainAvailability

以下代码示例演示了如何使用 CheckDomainAvailability。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CheckDomainAvailability中的。

CheckDomainTransferability

以下代码示例演示了如何使用 CheckDomainTransferability。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CheckDomainTransferability中的。

`GetDomainDetail`

以下代码示例演示了如何使用 `GetDomainDetail`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetDomainDetail中的。

GetDomainSuggestions

以下代码示例演示了如何使用 GetDomainSuggestions。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetDomainSuggestions中的。

`GetOperationDetail`

以下代码示例演示了如何使用 `GetOperationDetail`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetOperationDetail中的。

`ListDomains`

以下代码示例演示了如何使用 `ListDomains`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListDomains中的。

`ListOperations`

以下代码示例演示了如何使用 `ListOperations`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListOperations中的。

`ListPrices`

以下代码示例演示了如何使用 `ListPrices`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListPrices中的。

`RegisterDomain`

以下代码示例演示了如何使用 `RegisterDomain`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考RegisterDomain中的。

ViewBilling

以下代码示例演示了如何使用 ViewBilling。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ViewBilling中的。

使用 SDK for 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

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 基本功能
- 操作
- 场景
- 无服务器示例

开始使用

开始使用 Amazon S3

以下代码示例显示了如何开始使用 Amazon S3。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class HelloS3 {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBuckets(s3);
    }

    /**
     * Lists all the S3 buckets associated with the provided AWS S3 client.
     *
     * @param s3 the S3Client instance used to interact with the AWS S3 service
     */
    public static void listBuckets(S3Client s3) {
        try {
            ListBucketsResponse response = s3.listBuckets();
            List<Bucket> bucketList = response.buckets();
            bucketList.forEach(bucket -> {
                System.out.println("Bucket Name: " + bucket.name());
            });

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListBuckets中的。

基本功能

了解基本功能

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。

适用于 Java 的 SDK 2.x

> **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。

场景示例。

```
import java.io.IOException;
import java.util.Scanner;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * This Java code example performs the following tasks:
 *
 * 1. Creates an Amazon S3 bucket.
 * 2. Uploads an object to the bucket.
 * 3. Downloads the object to another local file.
 * 4. Uploads an object using multipart upload.
 * 5. List all objects located in the Amazon S3 bucket.
 * 6. Copies the object to another Amazon S3 bucket.
 * 7. Copy the object to another Amazon S3 bucket using multi copy.
 * 8. Deletes the object from the Amazon S3 bucket.
 * 9. Deletes the Amazon S3 bucket.
 */

public class S3Scenario {

    public static Scanner scanner = new Scanner(System.in);
    static S3Actions s3Actions = new S3Actions();
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final Logger logger = LoggerFactory.getLogger(S3Scenario.class);
    public static void main(String[] args) throws IOException {
        final String usage = """
            Usage:
               <bucketName> <key> <objectPath> <savePath> <toBucket>

            Where:
                bucketName - The name of the  S3 bucket.
                key - The unique identifier for the object stored in the S3 bucket.
                objectPath - The full file path of the object within the S3 bucket
 (e.g., "documents/reports/annual_report.pdf").
                savePath - The local file path where the object will be downloaded
 and saved (e.g., "C:/Users/username/Downloads/annual_report.pdf").
                toBucket - The name of the S3 bucket to which the object will be
 copied.
            """;

        if (args.length != 5) {
            logger.info(usage);
            return;
        }
```

```
        String bucketName = args[0];
        String key = args[1];
        String objectPath = args[2];
        String savePath = args[3];
        String toBucket = args[4];

        logger.info(DASHES);
        logger.info("Welcome to the Amazon Simple Storage Service (S3) example
scenario.");
        logger.info("""
            Amazon S3 is a highly scalable and durable object storage
            service provided by Amazon Web Services (AWS). It is designed to store
and retrieve
            any amount of data, from anywhere on the web, at any time.

            The `S3AsyncClient` interface in the AWS SDK for Java 2.x provides a set
of methods to
            programmatically interact with the Amazon S3 (Simple Storage Service)
service. This allows
            developers to automate the management and manipulation of S3 buckets and
objects as
            part of their application deployment pipelines. With S3, teams can focus
on building
            and deploying their applications without having to worry about the
underlying storage
            infrastructure required to host and manage large amounts of data.

            This scenario walks you through how to perform key operations for this
service.
            Let's get started...
            """);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        try {
            // Run the methods that belong to this scenario.
            runScenario(bucketName, key, objectPath, savePath, toBucket);

        } catch (Throwable rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception kmsEx) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
```

```
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
    }

    private static void runScenario(String bucketName, String key, String
objectPath, String savePath, String toBucket) throws Throwable {
        logger.info(DASHES);
        logger.info("1. Create an Amazon S3 bucket.");
        try {
            CompletableFuture<Void> future =
s3Actions.createBucketAsync(bucketName);
            future.join();
            waitForInputToContinue(scanner);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;

        }
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. Upload a local file to the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<PutObjectResponse> future =
s3Actions.uploadLocalFileAsync(bucketName, key, objectPath);
            future.join();
            logger.info("File uploaded successfully to {}/{}", bucketName, key);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
```

```
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);


        logger.info(DASHES);
        logger.info("3. Download the object to another local file.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.getObjectBytesAsync(bucketName, key, savePath);
            future.join();
            logger.info("Successfully obtained bytes from S3 object and wrote to
file {}", savePath);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("4. Perform a multipart upload.");
        waitForInputToContinue(scanner);
        String multipartKey = "multiPartKey";
        try {
            // Call the multipartUpload method
            CompletableFuture<Void> future = s3Actions.multipartUpload(bucketName,
multipartKey);
            future.join();
            logger.info("Multipart upload completed successfully for bucket '{}' and
key '{}'", bucketName, multipartKey);

        } catch (RuntimeException rt) {
```

```
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. List all objects located in the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.listAllObjectsAsync(bucketName);
            future.join();
            logger.info("Object listing completed successfully.");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. Copy the object to another Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
s3Actions.copyBucketObjectAsync(bucketName, key, toBucket);
            String result = future.join();
            logger.info("Copy operation result: {}", result);

        } catch (RuntimeException rt) {
```

```
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("7. Copy the object to another Amazon S3 bucket using multi
copy.");
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<String> future = s3Actions.performMultiCopy(toBucket,
bucketName, key);
            String result = future.join();
            logger.info("Copy operation result: {}", result);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("KMS error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);


        logger.info(DASHES);
        logger.info("8. Delete objects from the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.deleteObjectFromBucketAsync(bucketName, key);
            future.join();
```

```
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        try {
            CompletableFuture<Void> future =
s3Actions.deleteObjectFromBucketAsync(bucketName, "multiPartKey");
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("9. Delete the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.deleteBucketAsync(bucketName);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code {}",
s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
```

```
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("You successfully completed the Amazon S3 scenario.");
        logger.info(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                // Handle invalid input.
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

包含操作的包装器类。

```
public class S3Actions {

    private static final Logger logger = LoggerFactory.getLogger(S3Actions.class);
    private static S3AsyncClient s3AsyncClient;

    public static S3AsyncClient getAsyncClient() {
        if (s3AsyncClient == null) {
            /*
            The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
 version 2,
            and it is designed to provide a high-performance, asynchronous HTTP
 client for interacting with AWS services.
```

```
             It uses the Netty framework to handle the underlying network
communication and the Java NIO API to
             provide a non-blocking, event-driven approach to HTTP requests and
responses.
             */

            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(50)  // Adjust as needed.
                .connectionTimeout(Duration.ofSeconds(60))  // Set the connection
timeout.
                .readTimeout(Duration.ofSeconds(60))  // Set the read timeout.
                .writeTimeout(Duration.ofSeconds(60))  // Set the write timeout.
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))  // Set the overall API call
timeout.
                .apiCallAttemptTimeout(Duration.ofSeconds(90))  // Set the
individual call attempt timeout.
                .retryStrategy(RetryMode.STANDARD)
                .build();

            s3AsyncClient = S3AsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return s3AsyncClient;
    }


    /**
     * Creates an S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to create
     * @return a {@link CompletableFuture} that completes when the bucket is created
and ready
     * @throws RuntimeException if there is a failure while creating the bucket
     */
    public CompletableFuture<Void> createBucketAsync(String bucketName) {
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
```

```
            .build();

        CompletableFuture<CreateBucketResponse> response =
getAsyncClient().createBucket(bucketRequest);
        return response.thenCompose(resp -> {
            S3AsyncWaiter s3Waiter = getAsyncClient().waiter();
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();

            CompletableFuture<WaiterResponse<HeadBucketResponse>>
waiterResponseFuture =
                s3Waiter.waitUntilBucketExists(bucketRequestWait);
            return waiterResponseFuture.thenAccept(waiterResponse -> {
                waiterResponse.matched().response().ifPresent(headBucketResponse ->
{
                    logger.info(bucketName + " is ready");
                });
            });
        }).whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to create bucket", ex);
            }
        });
    }


    /**
     * Uploads a local file to an AWS S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to upload the file to
     * @param key        the key (object name) to use for the uploaded file
     * @param objectPath the local file path of the file to be uploaded
     * @return a {@link CompletableFuture} that completes with the {@link
PutObjectResponse} when the upload is successful, or throws a {@link
RuntimeException} if the upload fails
     */
    public CompletableFuture<PutObjectResponse> uploadLocalFileAsync(String
bucketName, String key, String objectPath) {
        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();
```

```
        CompletableFuture<PutObjectResponse> response =
getAsyncClient().putObject(objectRequest,
AsyncRequestBody.fromFile(Paths.get(objectPath)));
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to upload file", ex);
            }
        });
    }


    /**
     * Asynchronously retrieves the bytes of an object from an Amazon S3 bucket and
writes them to a local file.
     *
     * @param bucketName the name of the S3 bucket containing the object
     * @param keyName    the key (or name) of the S3 object to retrieve
     * @param path       the local file path where the object's bytes will be
written
     * @return a {@link CompletableFuture} that completes when the object bytes have
been written to the local file
     */
    public CompletableFuture<Void> getObjectBytesAsync(String bucketName, String
keyName, String path) {
        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        CompletableFuture<ResponseBytes<GetObjectResponse>> response =
getAsyncClient().getObject(objectRequest, AsyncResponseTransformer.toBytes());
        return response.thenAccept(objectBytes -> {
            try {
                byte[] data = objectBytes.asByteArray();
                Path filePath = Paths.get(path);
                Files.write(filePath, data);
                logger.info("Successfully obtained bytes from an S3 object");
            } catch (IOException ex) {
                throw new RuntimeException("Failed to write data to file", ex);
            }
        }).whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to get object bytes from S3",
ex);
```

```
            }
        });
    }


    /**
     * Asynchronously lists all objects in the specified S3 bucket.
     *
     * @param bucketName the name of the S3 bucket to list objects for
     * @return a {@link CompletableFuture} that completes when all objects have been
listed
     */
    public CompletableFuture<Void> listAllObjectsAsync(String bucketName) {
        ListObjectsV2Request initialRequest = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .maxKeys(1)
            .build();

        ListObjectsV2Publisher paginator =
getAsyncClient().listObjectsV2Paginator(initialRequest);
        return paginator.subscribe(response -> {
            response.contents().forEach(s3Object -> {
                logger.info("Object key: " + s3Object.key());
            });
        }).thenRun(() -> {
            logger.info("Successfully listed all objects in the bucket: " +
bucketName);
        }).exceptionally(ex -> {
            throw new RuntimeException("Failed to list objects", ex);
        });
    }


    /**
     * Asynchronously copies an object from one S3 bucket to another.
     *
     * @param fromBucket the name of the source S3 bucket
     * @param objectKey  the key (name) of the object to be copied
     * @param toBucket   the name of the destination S3 bucket
     * @return a {@link CompletableFuture} that completes with the copy result as a
{@link String}
     * @throws RuntimeException if the URL could not be encoded or an S3 exception
occurred during the copy
     */
```

```
   public CompletableFuture<String> copyBucketObjectAsync(String fromBucket, String
objectKey, String toBucket) {
       CopyObjectRequest copyReq = CopyObjectRequest.builder()
           .sourceBucket(fromBucket)
           .sourceKey(objectKey)
           .destinationBucket(toBucket)
           .destinationKey(objectKey)
           .build();

       CompletableFuture<CopyObjectResponse> response =
getAsyncClient().copyObject(copyReq);
       response.whenComplete((copyRes, ex) -> {
           if (copyRes != null) {
               logger.info("The " + objectKey + " was copied to " + toBucket);
           } else {
               throw new RuntimeException("An S3 exception occurred during copy",
ex);
           }
       });

       return response.thenApply(CopyObjectResponse::copyObjectResult)
           .thenApply(Object::toString);
   }

   /**
    * Performs a multipart upload to an Amazon S3 bucket.
    *
    * @param bucketName the name of the S3 bucket to upload the file to
    * @param key        the key (name) of the file to be uploaded
    * @return a {@link CompletableFuture} that completes when the multipart upload
is successful
    */
   public CompletableFuture<Void> multipartUpload(String bucketName, String key) {
       int mB = 1024 * 1024;

       CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
           .bucket(bucketName)
           .key(key)
           .build();

       return getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
           .thenCompose(createResponse -> {
               String uploadId = createResponse.uploadId();
```

```
                System.out.println("Upload ID: " + uploadId);

                // Upload part 1.
                UploadPartRequest uploadPartRequest1 = UploadPartRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .partNumber(1)
                    .contentLength((long) (5 * mB)) // Specify the content length
                    .build();

                CompletableFuture<CompletedPart> part1Future =
getAsyncClient().uploadPart(uploadPartRequest1,
                        AsyncRequestBody.fromByteBuffer(getRandomByteBuffer(5 *
mB)))
                    .thenApply(uploadPartResponse -> CompletedPart.builder()
                        .partNumber(1)
                        .eTag(uploadPartResponse.eTag())
                        .build());

                // Upload part 2.
                UploadPartRequest uploadPartRequest2 = UploadPartRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .partNumber(2)
                    .contentLength((long) (3 * mB))
                    .build();

                CompletableFuture<CompletedPart> part2Future =
getAsyncClient().uploadPart(uploadPartRequest2,
                        AsyncRequestBody.fromByteBuffer(getRandomByteBuffer(3 *
mB)))
                    .thenApply(uploadPartResponse -> CompletedPart.builder()
                        .partNumber(2)
                        .eTag(uploadPartResponse.eTag())
                        .build());

                // Combine the results of both parts.
                return CompletableFuture.allOf(part1Future, part2Future)
                    .thenCompose(v -> {
                        CompletedPart part1 = part1Future.join();
                        CompletedPart part2 = part2Future.join();
```

```
                        CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
                            .parts(part1, part2)
                            .build();

                        CompleteMultipartUploadRequest
completeMultipartUploadRequest = CompleteMultipartUploadRequest.builder()
                            .bucket(bucketName)
                            .key(key)
                            .uploadId(uploadId)
                            .multipartUpload(completedMultipartUpload)
                            .build();

                        // Complete the multipart upload
                        return
getAsyncClient().completeMultipartUpload(completeMultipartUploadRequest);
                    });
            })
            .thenAccept(response -> System.out.println("Multipart upload completed
successfully"))
            .exceptionally(ex -> {
                System.err.println("Failed to complete multipart upload: " +
ex.getMessage());
                throw new RuntimeException(ex);
            });
    }


    /**
     * Deletes an object from an S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket
     * @param key        the key (file name) of the object to be deleted
     * @return a {@link CompletableFuture} that completes when the object has been
deleted
     */
    public CompletableFuture<Void> deleteObjectFromBucketAsync(String bucketName,
String key) {
        DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();
```

```
        CompletableFuture<DeleteObjectResponse> response =
getAsyncClient().deleteObject(deleteObjectRequest);
        response.whenComplete((deleteRes, ex) -> {
            if (deleteRes != null) {
                logger.info(key + " was deleted");
            } else {
                throw new RuntimeException("An S3 exception occurred during delete",
ex);
            }
        });

        return response.thenApply(r -> null);
    }


    /**
     * Deletes an S3 bucket asynchronously.
     *
     * @param bucket the name of the bucket to be deleted
     * @return a {@link CompletableFuture} that completes when the bucket deletion
is successful, or throws a {@link RuntimeException}
     * if an error occurs during the deletion process
     */
    public CompletableFuture<Void> deleteBucketAsync(String bucket) {
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
            .bucket(bucket)
            .build();

        CompletableFuture<DeleteBucketResponse> response =
getAsyncClient().deleteBucket(deleteBucketRequest);
        response.whenComplete((deleteRes, ex) -> {
            if (deleteRes != null) {
                logger.info(bucket + " was deleted.");
            } else {
                throw new RuntimeException("An S3 exception occurred during bucket
deletion", ex);
            }
        });
        return response.thenApply(r -> null);
    }

    public CompletableFuture<String> performMultiCopy(String toBucket, String
bucketName, String key) {
```

```
        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
            .bucket(toBucket)
            .key(key)
            .build();

        getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
            .thenApply(createMultipartUploadResponse -> {
                String uploadId = createMultipartUploadResponse.uploadId();
                System.out.println("Upload ID: " + uploadId);

                UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
                    .sourceBucket(bucketName)
                    .destinationBucket(toBucket)
                    .sourceKey(key)
                    .destinationKey(key)
                    .uploadId(uploadId)  // Use the valid uploadId.
                    .partNumber(1)  // Ensure the part number is correct.
                    .copySourceRange("bytes=0-1023")  // Adjust range as needed
                    .build();

                return getAsyncClient().uploadPartCopy(uploadPartCopyRequest);
            })
            .thenCompose(uploadPartCopyFuture -> uploadPartCopyFuture)
            .whenComplete((uploadPartCopyResponse, exception) -> {
                if (exception != null) {
                    // Handle any exceptions.
                    logger.error("Error during upload part copy: " +
exception.getMessage());
                } else {
                    // Successfully completed the upload part copy.
                    System.out.println("Upload Part Copy completed successfully.
ETag: " + uploadPartCopyResponse.copyPartResult().eTag());
                }
            });
        return null;
    }

    private static ByteBuffer getRandomByteBuffer(int size) {
        ByteBuffer buffer = ByteBuffer.allocate(size);
        for (int i = 0; i < size; i++) {
            buffer.put((byte) (Math.random() * 256));
        }
```

```
        buffer.flip();
        return buffer;
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

操作

`AbortMultipartUpload`

以下代码示例演示了如何使用 `AbortMultipartUpload`。

适用于 Java 的 SDK 2.x

> **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。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
```

```
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.utils.builder.SdkBuilder;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.time.Duration;
import java.time.Instant;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;
import java.util.Objects;
import java.util.UUID;

import static software.amazon.awssdk.transfer.s3.SizeConstant.KB;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class AbortMultipartUploadExamples {
    static final String bucketName = "amzn-s3-demo-bucket" + UUID.randomUUID(); //
 Change bucket name.
    static final String key = UUID.randomUUID().toString();
```

```
    static final String classPathFilePath = "/multipartUploadFiles/s3-
userguide.pdf";
    static final String filePath = getFullFilePath(classPathFilePath);
    static final S3Client s3Client = S3Client.create();
    private static final Logger logger =
 LoggerFactory.getLogger(AbortMultipartUploadExamples.class);
    private static String accountId = getAccountId();

    public static void main(String[] args) {
        doAbortIncompleteMultipartUploadsFromList();
        doAbortMultipartUploadUsingUploadId();
        doAbortIncompleteMultipartUploadsOlderThan();
        doAbortMultipartUploadsUsingLifecycleConfig();
    }

    // A wrapper method that sets up the multipart upload environment for
 abortIncompleteMultipartUploadsFromList().
    public static void doAbortIncompleteMultipartUploadsFromList() {
        createBucket();
        initiateAndInterruptMultiPartUpload("uploadThread");
        abortIncompleteMultipartUploadsFromList();
        deleteResources();
    }

    /**
     * Aborts all incomplete multipart uploads from the specified S3 bucket.
     * <p>
     * This method retrieves a list of all incomplete multipart uploads in the
 specified S3 bucket,
     * and then aborts each of those uploads.
     */
    public static void abortIncompleteMultipartUploadsFromList() {
        ListMultipartUploadsRequest listMultipartUploadsRequest =
 ListMultipartUploadsRequest.builder()
            .bucket(bucketName)
            .build();

        ListMultipartUploadsResponse response =
 s3Client.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();

        AbortMultipartUploadRequest abortMultipartUploadRequest;
        for (MultipartUpload upload : uploads) {
            abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
```

```
                .bucket(bucketName)
                .key(upload.key())
                .expectedBucketOwner(accountId)
                .uploadId(upload.uploadId())
                .build();

            AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
            if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
upload.uploadId(), bucketName);
            }
        }
    }

    // A wrapper method that sets up the multipart upload environment for
abortIncompleteMultipartUploadsOlderThan().
    static void doAbortIncompleteMultipartUploadsOlderThan() {
        createBucket();
        Instant secondUploadInstant = initiateAndInterruptTwoUploads();
        abortIncompleteMultipartUploadsOlderThan(secondUploadInstant);
        deleteResources();
    }

    static void abortIncompleteMultipartUploadsOlderThan(Instant pointInTime) {
        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
            .bucket(bucketName)
            .build();

        ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();

        AbortMultipartUploadRequest abortMultipartUploadRequest;
        for (MultipartUpload upload : uploads) {
            logger.info("Found multipartUpload with upload ID [{}], initiated [{}]",
upload.uploadId(), upload.initiated());
            if (upload.initiated().isBefore(pointInTime)) {
                abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(upload.key())
                    .expectedBucketOwner(accountId)
                    .uploadId(upload.uploadId())
```

```
                    .build();

                AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
                if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                    logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
                }
            }
        }
    }

    // A wrapper method that sets up the multipart upload environment for
abortMultipartUploadUsingUploadId().
    static void doAbortMultipartUploadUsingUploadId() {
        createBucket();
        try {
            abortMultipartUploadUsingUploadId();
        } catch (S3Exception e) {
            logger.error(e.getMessage());
        } finally {
            deleteResources();
        }
    }

    static void abortMultipartUploadUsingUploadId() {
        String uploadId = startUploadReturningUploadId();
        AbortMultipartUploadResponse response = s3Client.abortMultipartUpload(b -> b
            .uploadId(uploadId)
            .bucket(bucketName)
            .key(key));

        if (response.sdkHttpResponse().isSuccessful()) {
            logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
uploadId, bucketName);
        }
    }

    // A wrapper method that sets up the multipart upload environment for
abortMultipartUploadsUsingLifecycleConfig().
    static void doAbortMultipartUploadsUsingLifecycleConfig() {
        createBucket();
        try {
            abortMultipartUploadsUsingLifecycleConfig();
```

```
        } catch (S3Exception e) {
            logger.error(e.getMessage());
        } finally {
            deleteResources();
        }
    }

    static void abortMultipartUploadsUsingLifecycleConfig() {
        Collection<LifecycleRule> lifeCycleRules = List.of(LifecycleRule.builder()
            .abortIncompleteMultipartUpload(b -> b.
                daysAfterInitiation(7))
            .status("Enabled")
            .filter(SdkBuilder::build) // Filter element is required.
            .build());

        // If the action is successful, the service sends back an HTTP 200 response
with an empty HTTP body.
        PutBucketLifecycleConfigurationResponse response =
s3Client.putBucketLifecycleConfiguration(b -> b
            .bucket(bucketName)
            .lifecycleConfiguration(b1 -> b1.rules(lifeCycleRules)));

        if (response.sdkHttpResponse().isSuccessful()) {
            logger.info("Rule to abort incomplete multipart uploads added to
bucket.");
        } else {
            logger.error("Unsuccessfully applied rule. HTTP status code is [{}]",
response.sdkHttpResponse().statusCode());
        }
    }

    /************************
     Multipart upload methods
     ************************/

    static void initiateAndInterruptMultiPartUpload(String threadName) {
        Runnable upload = () -> {
            try {
                AbortMultipartUploadExamples.doMultipartUpload();
            } catch (SdkException e) {
                logger.error(e.getMessage());
            }
        };
        Thread uploadThread = new Thread(upload, threadName);
```

```
        uploadThread.start();
        try {
            Thread.sleep(Duration.ofSeconds(1).toMillis()); // Give the multipart
upload time to register.
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
        }
        uploadThread.interrupt();
    }

    static Instant initiateAndInterruptTwoUploads() {
        Instant firstUploadInstant = Instant.now();
        initiateAndInterruptMultiPartUpload("uploadThread1");
        try {
            Thread.sleep(Duration.ofSeconds(5).toMillis());
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
        }
        Instant secondUploadInstant = Instant.now();
        initiateAndInterruptMultiPartUpload("uploadThread2");
        return secondUploadInstant;
    }

    static void doMultipartUpload() {
        String uploadId = step1CreateMultipartUpload();
        List<CompletedPart> completedParts = step2UploadParts(uploadId);
        step3CompleteMultipartUpload(uploadId, completedParts);
    }

    static String step1CreateMultipartUpload() {
        CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
            .bucket(bucketName)
            .key(key));
        return createMultipartUploadResponse.uploadId();
    }

    static List<CompletedPart> step2UploadParts(String uploadId) {
        int partNumber = 1;
        List<CompletedPart> completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(Long.valueOf(1024 * KB).intValue());

        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
```

```
            long position = 0;
            while (position < fileSize) {
                file.seek(position);
                long read = file.getChannel().read(bb);

                bb.flip(); // Swap position and limit before reading from the
buffer.
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .partNumber(partNumber)
                    .build();

                UploadPartResponse partResponse = s3Client.uploadPart(
                    uploadPartRequest,
                    RequestBody.fromByteBuffer(bb));

                CompletedPart part = CompletedPart.builder()
                    .partNumber(partNumber)
                    .eTag(partResponse.eTag())
                    .build();
                completedParts.add(part);
                logger.info("Part {} upload", partNumber);

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (IOException | S3Exception e) {
            logger.error(e.getMessage());
            return null;
        }
        return completedParts;
    }

    static void step3CompleteMultipartUpload(String uploadId, List<CompletedPart>
completedParts) {
        s3Client.completeMultipartUpload(b -> b
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
```

```
    }

    static String startUploadReturningUploadId() {
        String uploadId = step1CreateMultipartUpload();
        doMultipartUploadWithUploadId(uploadId);
        return uploadId;

    }

    static void doMultipartUploadWithUploadId(String uploadId) {
        new Thread(() -> {
            try {
                List<CompletedPart> completedParts = step2UploadParts(uploadId);
                step3CompleteMultipartUpload(uploadId, completedParts);
            } catch (SdkException e) {
                logger.error(e.getMessage());
            }
        }, "upload thread").start();
        try {
            Thread.sleep(Duration.ofSeconds(2L).toMillis());
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
            System.exit(1);
        }
    }

    /************************
     Resource handling methods
     ************************/

    static void createBucket() {
        logger.info("Creating bucket: [{}]", bucketName);
        s3Client.createBucket(b -> b.bucket(bucketName));
        try (S3Waiter s3Waiter = s3Client.waiter()) {
            s3Waiter.waitUntilBucketExists(b -> b.bucket(bucketName));
        }
        logger.info("Bucket created.");
    }

    static void deleteResources() {
        logger.info("Deleting resources ...");
        s3Client.deleteObject(b -> b.bucket(bucketName).key(key));
        s3Client.deleteBucket(b -> b.bucket(bucketName));
        try (S3Waiter s3Waiter = s3Client.waiter()) {
```

```
            s3Waiter.waitUntilBucketNotExists(b -> b.bucket(bucketName));
        }
        logger.info("Resources deleted.");
    }

    private static String getAccountId() {
        try (StsClient stsClient = StsClient.create()) {
            return stsClient.getCallerIdentity().account();
        }
    }

    static String getFullFilePath(String filePath) {
        URL uploadDirectoryURL = PerformMultiPartUpload.class.getResource(filePath);
        String fullFilePath;
        try {
            fullFilePath =
 Objects.requireNonNull(uploadDirectoryURL).toURI().getPath();
        } catch (URISyntaxException e) {
            throw new RuntimeException(e);
        }
        return fullFilePath;
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考AbortMultipartUpload中的。

`CopyObject`

以下代码示例演示了如何使用 `CopyObject`。

适用于 Java 的 SDK 2.x

> **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。

使用 S3Client 复制对象。

```
    /**
     * Asynchronously copies an object from one S3 bucket to another.
     *
     * @param fromBucket the name of the source S3 bucket
     * @param objectKey  the key (name) of the object to be copied
     * @param toBucket   the name of the destination S3 bucket
     * @return a {@link CompletableFuture} that completes with the copy result as a
{@link String}
     * @throws RuntimeException if the URL could not be encoded or an S3 exception
occurred during the copy
     */
    public CompletableFuture<String> copyBucketObjectAsync(String fromBucket, String
objectKey, String toBucket) {
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
            .sourceBucket(fromBucket)
            .sourceKey(objectKey)
            .destinationBucket(toBucket)
            .destinationKey(objectKey)
            .build();

        CompletableFuture<CopyObjectResponse> response =
getAsyncClient().copyObject(copyReq);
        response.whenComplete((copyRes, ex) -> {
            if (copyRes != null) {
                logger.info("The " + objectKey + " was copied to " + toBucket);
            } else {
                throw new RuntimeException("An S3 exception occurred during copy",
ex);
            }
        });

        return response.thenApply(CopyObjectResponse::copyObjectResult)
            .thenApply(Object::toString);
    }
```

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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CopyObject中的。

`CreateBucket`

以下代码示例演示了如何使用 `CreateBucket`。

适用于 Java 的 SDK 2.x

> **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。

创建存储桶。

```
   /**
    * Creates an S3 bucket asynchronously.
    *
    * @param bucketName the name of the S3 bucket to create
    * @return a {@link CompletableFuture} that completes when the bucket is created
and ready
    * @throws RuntimeException if there is a failure while creating the bucket
    */
   public CompletableFuture<Void> createBucketAsync(String bucketName) {
       CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
           .bucket(bucketName)
           .build();

       CompletableFuture<CreateBucketResponse> response =
getAsyncClient().createBucket(bucketRequest);
       return response.thenCompose(resp -> {
           S3AsyncWaiter s3Waiter = getAsyncClient().waiter();
           HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
               .bucket(bucketName)
               .build();

           CompletableFuture<WaiterResponse<HeadBucketResponse>>
waiterResponseFuture =
               s3Waiter.waitUntilBucketExists(bucketRequestWait);
           return waiterResponseFuture.thenAccept(waiterResponse -> {
               waiterResponse.matched().response().ifPresent(headBucketResponse ->
{
                   logger.info(bucketName + " is ready");
               });
           });
       }).whenComplete((resp, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to create bucket", ex);
           }
       });
   }
```

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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateBucket中的。

`DeleteBucket`

以下代码示例演示了如何使用 `DeleteBucket`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Deletes an S3 bucket asynchronously.
     *
     * @param bucket the name of the bucket to be deleted
     * @return a {@link CompletableFuture} that completes when the bucket deletion
is successful, or throws a {@link RuntimeException}
     * if an error occurs during the deletion process
     */
```

```
    public CompletableFuture<Void> deleteBucketAsync(String bucket) {
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
            .bucket(bucket)
            .build();

        CompletableFuture<DeleteBucketResponse> response =
getAsyncClient().deleteBucket(deleteBucketRequest);
        response.whenComplete((deleteRes, ex) -> {
            if (deleteRes != null) {
                logger.info(bucket + " was deleted.");
            } else {
                throw new RuntimeException("An S3 exception occurred during bucket
deletion", ex);
            }
        });
        return response.thenApply(r -> null);
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteBucket中的。

`DeleteBucketPolicy`

以下代码示例演示了如何使用 `DeleteBucketPolicy`。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteBucketPolicy {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <bucketName>

                Where:
                    bucketName - The Amazon S3 bucket to delete the policy from (for
 example, bucket1).""";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Deleting policy from bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();

        deleteS3BucketPolicy(s3, bucketName);
        s3.close();
    }

    /**
     * Deletes the S3 bucket policy for the specified bucket.
     *
     * @param s3 the {@link S3Client} instance to use for the operation
     * @param bucketName the name of the S3 bucket for which the policy should be
 deleted
     *
     * @throws S3Exception if there is an error deleting the bucket policy
     */
    public static void deleteS3BucketPolicy(S3Client s3, String bucketName) {
        DeleteBucketPolicyRequest delReq = DeleteBucketPolicyRequest.builder()
```

```
                .bucket(bucketName)
                .build();

        try {
            s3.deleteBucketPolicy(delReq);
            System.out.println("Done!");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteBucketPolicy中的。

`DeleteBucketWebsite`

以下代码示例演示了如何使用 `DeleteBucketWebsite`。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
 */

public class DeleteWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = """

            Usage:     <bucketName>

            Where:
                bucketName - The Amazon S3 bucket to delete the website
 configuration from.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Deleting website configuration for Amazon S3 bucket: %s
\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteBucketWebsiteConfig(s3, bucketName);
        System.out.println("Done!");
        s3.close();
    }

    /**
     * Deletes the website configuration for an Amazon S3 bucket.
     *
     * @param s3 The {@link S3Client} instance used to interact with Amazon S3.
     * @param bucketName The name of the S3 bucket for which the website
 configuration should be deleted.
     * @throws S3Exception If an error occurs while deleting the website
 configuration.
     */
    public static void deleteBucketWebsiteConfig(S3Client s3, String bucketName) {
        DeleteBucketWebsiteRequest delReq = DeleteBucketWebsiteRequest.builder()
            .bucket(bucketName)
            .build();
```

```
        try {
            s3.deleteBucketWebsite(delReq);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.out.println("Failed to delete website configuration!");
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteBucketWebsite中的。

`DeleteObject`

以下代码示例演示了如何使用 `DeleteObject`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Deletes an object from an S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket
     * @param key        the key (file name) of the object to be deleted
     * @return a {@link CompletableFuture} that completes when the object has been
deleted
     */
    public CompletableFuture<Void> deleteObjectFromBucketAsync(String bucketName,
String key) {
        DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();
```

```
        CompletableFuture<DeleteObjectResponse> response =
getAsyncClient().deleteObject(deleteObjectRequest);
        response.whenComplete((deleteRes, ex) -> {
            if (deleteRes != null) {
                logger.info(key + " was deleted");
            } else {
                throw new RuntimeException("An S3 exception occurred during delete",
ex);
            }
        });

        return response.thenApply(r -> null);
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteObject中的。

`DeleteObjects`

以下代码示例演示了如何使用 `DeleteObjects`。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.ArrayList;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteMultiObjects {
    public static void main(String[] args) {
        final String usage = """

            Usage:    <bucketName>

            Where:
               bucketName - the Amazon S3 bucket name.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteBucketObjects(s3, bucketName);
        s3.close();
    }

    /**
     * Deletes multiple objects from an Amazon S3 bucket.
     *
     * @param s3 An Amazon S3 client object.
     * @param bucketName The name of the Amazon S3 bucket to delete objects from.
     */
    public static void deleteBucketObjects(S3Client s3, String bucketName) {
        // Upload three sample objects to the specfied Amazon S3 bucket.
        ArrayList<ObjectIdentifier> keys = new ArrayList<>();
        PutObjectRequest putOb;
```

```
        ObjectIdentifier objectId;

        for (int i = 0; i < 3; i++) {
            String keyName = "delete object example " + i;
            objectId = ObjectIdentifier.builder()
                .key(keyName)
                .build();

            putOb = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            s3.putObject(putOb, RequestBody.fromString(keyName));
            keys.add(objectId);
        }

        System.out.println(keys.size() + " objects successfully created.");

        // Delete multiple objects in one request.
        Delete del = Delete.builder()
            .objects(keys)
            .build();

        try {
            DeleteObjectsRequest multiObjectDeleteRequest =
 DeleteObjectsRequest.builder()
                .bucket(bucketName)
                .delete(del)
                .build();

            s3.deleteObjects(multiObjectDeleteRequest);
            System.out.println("Multiple objects are deleted!");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteObjects中的。

GetBucketAcl

以下代码示例演示了如何使用 GetBucketAcl。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAclRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAclResponse;
import software.amazon.awssdk.services.s3.model.Grant;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetAcl {
    public static void main(String[] args) {
        final String usage = """

            Usage:
              <bucketName> <objectKey>

            Where:
              bucketName - The Amazon S3 bucket to get the access control list (ACL)
 for.
              objectKey - The object to get the ACL for.\s
            """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        System.out.println("Retrieving ACL for object: " + objectKey);
        System.out.println("in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getBucketACL(s3, objectKey, bucketName);
        s3.close();
        System.out.println("Done!");
    }

    /**
     * Retrieves the Access Control List (ACL) for an object in an Amazon S3 bucket.
     *
     * @param s3 The S3Client object used to interact with the Amazon S3 service.
     * @param objectKey The key of the object for which the ACL is to be retrieved.
     * @param bucketName The name of the bucket containing the object.
     * @return The ID of the grantee who has permission on the object, or an empty
string if an error occurs.
     */
    public static String getBucketACL(S3Client s3, String objectKey, String
bucketName) {
        try {
            GetObjectAclRequest aclReq = GetObjectAclRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectAclResponse aclRes = s3.getObjectAcl(aclReq);
            List<Grant> grants = aclRes.grants();
            String grantee = "";
            for (Grant grant : grants) {
                System.out.format("  %s: %s\n", grant.grantee().id(),
grant.permission());
                grantee = grant.grantee().id();
```

```
            }

            return grantee;
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetBucketAcl中的。

`GetBucketPolicy`

以下代码示例演示了如何使用 `GetBucketPolicy`。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class GetBucketPolicy {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName>

            Where:
                bucketName - The Amazon S3 bucket to get the policy from.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        String polText = getPolicy(s3, bucketName);
        System.out.println("Policy Text: " + polText);
        s3.close();
    }

    /**
     * Retrieves the policy for the specified Amazon S3 bucket.
     *
     * @param s3 the {@link S3Client} instance to use for making the request
     * @param bucketName the name of the S3 bucket for which to retrieve the policy
     * @return the policy text for the specified bucket, or an empty string if an
 error occurs
     */
    public static String getPolicy(S3Client s3, String bucketName) {
        String policyText;
        System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
        GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
            .bucket(bucketName)
            .build();
```

```
        try {
            GetBucketPolicyResponse policyRes = s3.getBucketPolicy(policyReq);
            policyText = policyRes.policy();
            return policyText;

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[GetBucketPolicy](#)中的。

`GetBucketReplication`

以下代码示例演示了如何使用 `GetBucketReplication`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Retrieves the replication details for the specified S3 bucket.
     *
     * @param s3Client           the S3 client used to interact with the S3 service
     * @param sourceBucketName   the name of the S3 bucket to retrieve the
replication details for
     *
     * @throws S3Exception if there is an error retrieving the replication details
     */
    public static void getReplicationDetails(S3Client s3Client, String
sourceBucketName) {
        GetBucketReplicationRequest getRequest =
GetBucketReplicationRequest.builder()
```

```
            .bucket(sourceBucketName)
            .build();

        try {
            ReplicationConfiguration replicationConfig =
s3Client.getBucketReplication(getRequest).replicationConfiguration();
            ReplicationRule rule = replicationConfig.rules().get(0);
            System.out.println("Retrieved destination bucket: " +
rule.destination().bucket());
            System.out.println("Retrieved priority: " + rule.priority());
            System.out.println("Retrieved source-bucket replication rule status: " +
rule.status());

        } catch (S3Exception e) {
            System.err.println("Failed to retrieve replication details: " +
e.awsErrorDetails().errorMessage());
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetBucketReplication中的。

`GetObject`

以下代码示例演示了如何使用 `GetObject`。

适用于 Java 的 SDK 2.x

> **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。

使用 S3Client 以字节数组读取数据。

```
    /**
     * Asynchronously retrieves the bytes of an object from an Amazon S3 bucket and
writes them to a local file.
     *
     * @param bucketName the name of the S3 bucket containing the object
```

```
    * @param keyName    the key (or name) of the S3 object to retrieve
    * @param path       the local file path where the object's bytes will be
written
    * @return a {@link CompletableFuture} that completes when the object bytes have
been written to the local file
    */
   public CompletableFuture<Void> getObjectBytesAsync(String bucketName, String
keyName, String path) {
       GetObjectRequest objectRequest = GetObjectRequest.builder()
           .key(keyName)
           .bucket(bucketName)
           .build();

       CompletableFuture<ResponseBytes<GetObjectResponse>> response =
getAsyncClient().getObject(objectRequest, AsyncResponseTransformer.toBytes());
       return response.thenAccept(objectBytes -> {
           try {
               byte[] data = objectBytes.asByteArray();
               Path filePath = Paths.get(path);
               Files.write(filePath, data);
               logger.info("Successfully obtained bytes from an S3 object");
           } catch (IOException ex) {
               throw new RuntimeException("Failed to write data to file", ex);
           }
       }).whenComplete((resp, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to get object bytes from S3",
ex);
           }
       });
   }
```

使用 S 3 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.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
```

```
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.NoSuchFileException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;

    public Long downloadFile(S3TransferManager transferManager, String bucketName,
                             String key, String downloadedFileWithPath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
                .getObjectRequest(b -> b.bucket(bucketName).key(key))
                .destination(Paths.get(downloadedFileWithPath))
                .build();

        FileDownload downloadFile =
 transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult =
 downloadFile.completionFuture().join();
        logger.info("Content length [{}]",
 downloadResult.response().contentLength());
        return downloadResult.response().contentLength();
    }
```

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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectTags {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents the object.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listTags(s3, bucketName, keyName);
        s3.close();
    }

    /**
     * Lists the tags associated with an Amazon S3 object.
     *
     * @param s3 the S3Client object used to interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket that contains the object
     * @param keyName the key (name) of the S3 object
     */
    public static void listTags(S3Client s3, String bucketName, String keyName) {
```

```
        try {
            GetObjectTaggingRequest getTaggingRequest = GetObjectTaggingRequest
                .builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            GetObjectTaggingResponse tags = s3.getObjectTagging(getTaggingRequest);
            List<Tag> tagSet = tags.tagSet();
            for (Tag tag : tagSet) {
                System.out.println(tag.key());
                System.out.println(tag.value());
            }

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

使用 S3Client 获取对象的 URL。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetUrlRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.net.URL;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectUrl {
    public static void main(String[] args) {
```

```
        final String usage = """

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.
                keyName - A key name that represents the object.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getURL(s3, bucketName, keyName);
        s3.close();
    }

    /**
     * Retrieves the URL for a specific object in an Amazon S3 bucket.
     *
     * @param s3 the S3Client object used to interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket where the object is stored
     * @param keyName the name of the object for which the URL should be retrieved
     * @throws S3Exception if there is an error retrieving the URL for the specified
object
     */
    public static void getURL(S3Client s3, String bucketName, String keyName) {
        try {
            GetUrlRequest request = GetUrlRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            URL url = s3.utilities().getUrl(request);
            System.out.println("The URL for  " + keyName + " is " + url);
```

```
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetObjectPresignedUrl {
    public static void main(String[] args) {
        final String USAGE = """

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
```

```
                keyName - A key name that represents a text file.\s
            """;

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Presigner presigner = S3Presigner.builder()
            .region(region)
            .build();

        getPresignedUrl(presigner, bucketName, keyName);
        presigner.close();
    }

    /**
     * Generates a pre-signed URL for an Amazon S3 object.
     *
     * @param presigner The {@link S3Presigner} instance to use for generating the
pre-signed URL.
     * @param bucketName The name of the Amazon S3 bucket where the object is
stored.
     * @param keyName The key name (file name) of the object in the Amazon S3
bucket.
     *
     * @throws S3Exception If there is an error interacting with the Amazon S3
service.
     * @throws IOException If there is an error opening the HTTP connection or
reading/writing the request/response.
     */
    public static void getPresignedUrl(S3Presigner presigner, String bucketName,
String keyName) {
        try {
            GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            GetObjectPresignRequest getObjectPresignRequest =
GetObjectPresignRequest.builder()
```

```
                .signatureDuration(Duration.ofMinutes(60))
                .getObjectRequest(getObjectRequest)
                .build();

            PresignedGetObjectRequest presignedGetObjectRequest =
 presigner.presignGetObject(getObjectPresignRequest);
            String theUrl = presignedGetObjectRequest.url().toString();
            System.out.println("Presigned URL: " + theUrl);
            HttpURLConnection connection = (HttpURLConnection)
 presignedGetObjectRequest.url().openConnection();
            presignedGetObjectRequest.httpRequest().headers().forEach((header,
 values) -> {
                values.forEach(value -> {
                    connection.addRequestProperty(header, value);
                });
            });

            // Send any request payload that the service needs (not needed when
            // isBrowserExecutable is true).
            if (presignedGetObjectRequest.signedPayload().isPresent()) {
                connection.setDoOutput(true);

                try (InputStream signedPayload =
 presignedGetObjectRequest.signedPayload().get().asInputStream();
                     OutputStream httpOutputStream = connection.getOutputStream()) {
                    IoUtils.copy(signedPayload, httpOutputStream);
                }
            }

            // Download the result of executing the request.
            try (InputStream content = connection.getInputStream()) {
                System.out.println("Service returned response: ");
                IoUtils.copy(content, System.out);
            }

        } catch (S3Exception | IOException e) {
            e.printStackTrace();
        }
    }
}
```

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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectData {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <keyName> <path>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
                path - The path where the file is written to.\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
```

```
        String path = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getObjectBytes(s3, bucketName, keyName, path);
        s3.close();
    }

    /**
     * Retrieves the bytes of an object stored in an Amazon S3 bucket and saves them
to a local file.
     *
     * @param s3 The S3Client instance used to interact with the Amazon S3 service.
     * @param bucketName The name of the S3 bucket where the object is stored.
     * @param keyName The key (or name) of the S3 object.
     * @param path The local file path where the object's bytes will be saved.
     * @throws IOException If an I/O error occurs while writing the bytes to the
local file.
     * @throws S3Exception If an error occurs while retrieving the object from the
S3 bucket.
     */
    public static void getObjectBytes(S3Client s3, String bucketName, String
keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                .builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            ResponseBytes<GetObjectResponse> objectBytes =
s3.getObject(objectRequest, ResponseTransformer.toBytes());
            byte[] data = objectBytes.asByteArray();

            // Write the data to a local file.
            File myFile = new File(path);
            OutputStream os = new FileOutputStream(myFile);
            os.write(data);
            System.out.println("Successfully obtained bytes from an S3 object");
            os.close();

        } catch (IOException ex) {
```

```
            ex.printStackTrace();
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[GetObject](#)中的。

`GetObjectLegalHold`

以下代码示例演示了如何使用 `GetObjectLegalHold`。

适用于 Java 的 SDK 2.x

> **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。

```
   // 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetObjectLegalHold中的。

`GetObjectLockConfiguration`

以下代码示例演示了如何使用 `GetObjectLockConfiguration`。

适用于 Java 的 SDK 2.x

> **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。

```
    // Get the object lock configuration details for an S3 bucket.
    public void getBucketObjectLockConfiguration(String bucketName) {
        GetObjectLockConfigurationRequest objectLockConfigurationRequest =
GetObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .build();

        GetObjectLockConfigurationResponse response =
getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
        System.out.println("Bucket object lock config for "+bucketName +":  ");
        System.out.println("\tEnabled:
"+response.objectLockConfiguration().objectLockEnabled());
        System.out.println("\tRule: "+
response.objectLockConfiguration().rule().defaultRetention());
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetObjectLockConfiguration中的。

GetObjectRetention

以下代码示例演示了如何使用 GetObjectRetention。

适用于 Java 的 SDK 2.x

> **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。

```
   // 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;
       }
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetObjectRetention中的。

HeadObject

以下代码示例演示了如何使用 HeadObject。

适用于 Java 的 SDK 2.x

> **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。

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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetObjectContentType {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <keyName>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
```

```
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getContentType(s3, bucketName, keyName);
        s3.close();
    }

    /**
     * Retrieves the content type of an object stored in an Amazon S3 bucket.
     *
     * @param s3 an instance of the {@link S3Client} class, which is used to
 interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket where the object is stored
     * @param keyName the key (file name) of the object in the S3 bucket
     */
    public static void getContentType(S3Client s3, String bucketName, String
 keyName) {
        try {
            HeadObjectRequest objectRequest = HeadObjectRequest.builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            HeadObjectResponse objectHead = s3.headObject(objectRequest);
            String type = objectHead.contentType();
            System.out.println("The object content type is " + type);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

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();
    }

    /**
     * Checks the restoration status of an Amazon S3 object.
     *
     * @param s3         an instance of the {@link S3Client} class used to interact
 with the Amazon S3 service
     * @param bucketName the name of the Amazon S3 bucket where the object is stored
     * @param keyName    the name of the Amazon S3 object to be checked
     * @throws S3Exception if an error occurs while interacting with the Amazon S3
 service
     */
    public static void checkStatus(S3Client s3, String bucketName, String keyName) {
        try {
```

```
            HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            HeadObjectResponse response = s3.headObject(headObjectRequest);
            System.out.println("The Amazon S3 object restoration status is " +
 response.restore());

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考HeadObject中的。

`ListBuckets`

以下代码示例演示了如何使用 `ListBuckets`。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListBucketsIterable;
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class ListBuckets {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listAllBuckets(s3);

    }

    /**
     * Lists all the S3 buckets available in the current AWS account.
     *
     * @param s3 The {@link S3Client} instance to use for interacting with the
 Amazon S3 service.
     */
    public static void listAllBuckets(S3Client s3) {
        ListBucketsIterable response = s3.listBucketsPaginator();
        response.buckets().forEach(bucket ->
            System.out.println("Bucket Name: " + bucket.name()));
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListBuckets中的。

`ListMultipartUploads`

以下代码示例演示了如何使用 `ListMultipartUploads`。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
```

```
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class ListMultipartUploads {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <bucketName>\s

                Where:
                    bucketName - The name of the Amazon S3 bucket where an in-
progress multipart upload is occurring.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
                .region(region)
                .build();
        listUploads(s3, bucketName);
        s3.close();
    }

    /**
     * Lists the multipart uploads currently in progress in the specified Amazon S3
 bucket.
     *
     * @param s3 the S3Client object used to interact with Amazon S3
```

```
     * @param bucketName the name of the Amazon S3 bucket to list the multipart
 uploads for
     */
    public static void listUploads(S3Client s3, String bucketName) {
        try {
            ListMultipartUploadsRequest listMultipartUploadsRequest =
 ListMultipartUploadsRequest.builder()
                    .bucket(bucketName)
                    .build();

            ListMultipartUploadsResponse response =
 s3.listMultipartUploads(listMultipartUploadsRequest);
            List<MultipartUpload> uploads = response.uploads();
            for (MultipartUpload upload : uploads) {
                System.out.println("Upload in progress: Key = \"" + upload.key() +
 "\", id = " + upload.uploadId());
            }

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListMultipartUploads中的。

`ListObjectsV2`

以下代码示例演示了如何使用 `ListObjectsV2`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
```

```
     * Asynchronously lists all objects in the specified S3 bucket.
     *
     * @param bucketName the name of the S3 bucket to list objects for
     * @return a {@link CompletableFuture} that completes when all objects have been
listed
     */
    public CompletableFuture<Void> listAllObjectsAsync(String bucketName) {
        ListObjectsV2Request initialRequest = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .maxKeys(1)
            .build();

        ListObjectsV2Publisher paginator =
getAsyncClient().listObjectsV2Paginator(initialRequest);
        return paginator.subscribe(response -> {
            response.contents().forEach(s3Object -> {
                logger.info("Object key: " + s3Object.key());
            });
        }).thenRun(() -> {
            logger.info("Successfully listed all objects in the bucket: " +
bucketName);
        }).exceptionally(ex -> {
            throw new RuntimeException("Failed to list objects", ex);
        });
    }
```

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();
    }

    /**
     * Lists the objects in the specified S3 bucket.
     *
     * @param s3 the S3Client instance used to interact with Amazon S3
     * @param bucketName the name of the S3 bucket to list the objects from
     */
    public static void listBucketObjects(S3Client s3, String bucketName) {
        try {
            ListObjectsV2Request listReq = ListObjectsV2Request.builder()
                .bucket(bucketName)
                .maxKeys(1)
                .build();

            ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);
            listRes.stream()
                .flatMap(r -> r.contents().stream())
                .forEach(content -> System.out.println(" Key: " + content.key() + "
 size = " + content.size()));

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考中的 ListObjectsV2。

PutBucketAcl

以下代码示例演示了如何使用 PutBucketAcl。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AccessControlPolicy;
import software.amazon.awssdk.services.s3.model.Grant;
import software.amazon.awssdk.services.s3.model.Permission;
import software.amazon.awssdk.services.s3.model.PutBucketAclRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Type;

import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetAcl {
    public static void main(String[] args) {
        final String usage = """

            Usage:
```

```
              <bucketName> <id>\s

            Where:
              bucketName - The Amazon S3 bucket to grant permissions on.\s
              id - The ID of the owner of this bucket (you can get this value from
the AWS Management Console).
            """;

        if (args.length != 2) {
            System.out.println(usage);
            return;
        }

        String bucketName = args[0];
        String id = args[1];
        System.out.format("Setting access \n");
        System.out.println(" in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setBucketAcl(s3, bucketName, id);
        System.out.println("Done!");
        s3.close();
    }

    /**
     * Sets the Access Control List (ACL) for an Amazon S3 bucket.
     *
     * @param s3 the S3Client instance to be used for the operation
     * @param bucketName the name of the S3 bucket to set the ACL for
     * @param id the ID of the AWS user or account that will be granted full control
of the bucket
     * @throws S3Exception if an error occurs while setting the bucket ACL
     */
    public static void setBucketAcl(S3Client s3, String bucketName, String id) {
        try {
            Grant ownerGrant = Grant.builder()
                .grantee(builder -> builder.id(id)
                    .type(Type.CANONICAL_USER))
                .permission(Permission.FULL_CONTROL)
                .build();
```

```
            List<Grant> grantList2 = new ArrayList<>();
            grantList2.add(ownerGrant);

            AccessControlPolicy acl = AccessControlPolicy.builder()
                .owner(builder -> builder.id(id))
                .grants(grantList2)
                .build();

            PutBucketAclRequest putAclReq = PutBucketAclRequest.builder()
                .bucket(bucketName)
                .accessControlPolicy(acl)
                .build();

            s3.putBucketAcl(putAclReq);

        } catch (S3Exception e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutBucketAcl中的。

`PutBucketCors`

以下代码示例演示了如何使用 `PutBucketCors`。

适用于 Java 的 SDK 2.x

> **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。

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

import java.util.ArrayList;
```

```
import java.util.List;

import software.amazon.awssdk.services.s3.model.GetBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.GetBucketCorsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.CORSRule;
import software.amazon.awssdk.services.s3.model.CORSConfiguration;
import software.amazon.awssdk.services.s3.model.PutBucketCorsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class S3Cors {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <accountId>\s

            Where:
                bucketName - The Amazon S3 bucket to upload an object into.
                accountId - The id of the account that owns the Amazon S3 bucket.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String accountId = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setCorsInformation(s3, bucketName, accountId);
        getBucketCorsInformation(s3, bucketName, accountId);
```

```
        deleteBucketCorsInformation(s3, bucketName, accountId);
        s3.close();
    }

    /**
     * Deletes the CORS (Cross-Origin Resource Sharing) configuration for an Amazon
S3 bucket.
     *
     * @param s3            the {@link S3Client} instance used to interact with the
Amazon S3 service
     * @param bucketName    the name of the Amazon S3 bucket for which the CORS
configuration should be deleted
     * @param accountId     the expected AWS account ID of the bucket owner
     *
     * @throws S3Exception if an error occurs while deleting the CORS configuration
for the bucket
     */
    public static void deleteBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
        try {
            DeleteBucketCorsRequest bucketCorsRequest =
DeleteBucketCorsRequest.builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            s3.deleteBucketCors(bucketCorsRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Retrieves the CORS (Cross-Origin Resource Sharing) configuration for the
specified S3 bucket.
     *
     * @param s3 the S3Client instance to use for the operation
     * @param bucketName the name of the S3 bucket to retrieve the CORS
configuration for
     * @param accountId the expected bucket owner's account ID
     *
     * @throws S3Exception if there is an error retrieving the CORS configuration
```

```
     */
    public static void getBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
        try {
            GetBucketCorsRequest bucketCorsRequest = GetBucketCorsRequest.builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            GetBucketCorsResponse corsResponse =
s3.getBucketCors(bucketCorsRequest);
            List<CORSRule> corsRules = corsResponse.corsRules();
            for (CORSRule rule : corsRules) {
                System.out.println("allowOrigins: " + rule.allowedOrigins());
                System.out.println("AllowedMethod: " + rule.allowedMethods());
            }

        } catch (S3Exception e) {

            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Sets the Cross-Origin Resource Sharing (CORS) rules for an Amazon S3 bucket.
     *
     * @param s3 The S3Client object used to interact with the Amazon S3 service.
     * @param bucketName The name of the S3 bucket to set the CORS rules for.
     * @param accountId The AWS account ID of the bucket owner.
     */
    public static void setCorsInformation(S3Client s3, String bucketName, String
accountId) {
        List<String> allowMethods = new ArrayList<>();
        allowMethods.add("PUT");
        allowMethods.add("POST");
        allowMethods.add("DELETE");

        List<String> allowOrigins = new ArrayList<>();
        allowOrigins.add("http://example.com");
        try {
            // Define CORS rules.
            CORSRule corsRule = CORSRule.builder()
                .allowedMethods(allowMethods)
```

```
                .allowedOrigins(allowOrigins)
                .build();

            List<CORSRule> corsRules = new ArrayList<>();
            corsRules.add(corsRule);
            CORSConfiguration configuration = CORSConfiguration.builder()
                .corsRules(corsRules)
                .build();

            PutBucketCorsRequest putBucketCorsRequest =
 PutBucketCorsRequest.builder()
                .bucket(bucketName)
                .corsConfiguration(configuration)
                .expectedBucketOwner(accountId)
                .build();

            s3.putBucketCors(putBucketCorsRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutBucketCors中的。

`PutBucketLifecycleConfiguration`

以下代码示例演示了如何使用 `PutBucketLifecycleConfiguration`。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.Transition;
import
 software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationRequest;
import
 software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketLifecycleRequest;
import software.amazon.awssdk.services.s3.model.TransitionStorageClass;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class LifecycleConfiguration {
    public static void main(String[] args) {
        final String usage = """

            Usage:
              <bucketName> <accountId>\s

            Where:
              bucketName - The Amazon Simple Storage Service (Amazon S3) bucket to
 upload an object into.
              accountId - The id of the account that owns the Amazon S3 bucket.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String bucketName = args[0];
        String accountId = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setLifecycleConfig(s3, bucketName, accountId);
        getLifecycleConfig(s3, bucketName, accountId);
        deleteLifecycleConfig(s3, bucketName, accountId);
        System.out.println("You have successfully created, updated, and deleted a
Lifecycle configuration");
        s3.close();
    }

    /**
     * Sets the lifecycle configuration for an Amazon S3 bucket.
     *
     * @param s3           The Amazon S3 client to use for the operation.
     * @param bucketName   The name of the Amazon S3 bucket.
     * @param accountId    The expected owner of the Amazon S3 bucket.
     *
     * @throws S3Exception if there is an error setting the lifecycle configuration.
     */
    public static void setLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
        try {
            // Create a rule to archive objects with the "glacierobjects/" prefix to
the
            // S3 Glacier Flexible Retrieval storage class immediately.
            LifecycleRuleFilter ruleFilter = LifecycleRuleFilter.builder()
                .prefix("glacierobjects/")
                .build();

            Transition transition = Transition.builder()
                .storageClass(TransitionStorageClass.GLACIER)
                .days(0)
                .build();

            LifecycleRule rule1 = LifecycleRule.builder()
                .id("Archive immediately rule")
                .filter(ruleFilter)
                .transitions(transition)
```

```
                .status(ExpirationStatus.ENABLED)
                .build();

            // Create a second rule.
            Transition transition2 = Transition.builder()
                .storageClass(TransitionStorageClass.GLACIER)
                .days(0)
                .build();

            List<Transition> transitionList = new ArrayList<>();
            transitionList.add(transition2);

            LifecycleRuleFilter ruleFilter2 = LifecycleRuleFilter.builder()
                .prefix("glacierobjects/")
                .build();

            LifecycleRule rule2 = LifecycleRule.builder()
                .id("Archive and then delete rule")
                .filter(ruleFilter2)
                .transitions(transitionList)
                .status(ExpirationStatus.ENABLED)
                .build();

            // Add the LifecycleRule objects to an ArrayList.
            ArrayList<LifecycleRule> ruleList = new ArrayList<>();
            ruleList.add(rule1);
            ruleList.add(rule2);

            BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
                .rules(ruleList)
                .build();

            PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
                .builder()
                .bucket(bucketName)
                .lifecycleConfiguration(lifecycleConfiguration)
                .expectedBucketOwner(accountId)
                .build();


s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);
```

```
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Retrieves the lifecycle configuration for an Amazon S3 bucket and adds a new
lifecycle rule to it.
     *
     * @param s3 the S3Client instance used to interact with Amazon S3
     * @param bucketName the name of the Amazon S3 bucket
     * @param accountId the expected owner of the Amazon S3 bucket
     */
    public static void getLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
        try {
            GetBucketLifecycleConfigurationRequest
getBucketLifecycleConfigurationRequest = GetBucketLifecycleConfigurationRequest
                .builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            GetBucketLifecycleConfigurationResponse response = s3

.getBucketLifecycleConfiguration(getBucketLifecycleConfigurationRequest);
            List<LifecycleRule> newList = new ArrayList<>();
            List<LifecycleRule> rules = response.rules();
            for (LifecycleRule rule : rules) {
                newList.add(rule);
            }

            // Add a new rule with both a prefix predicate and a tag predicate.
            LifecycleRuleFilter ruleFilter = LifecycleRuleFilter.builder()
                .prefix("YearlyDocuments/")
                .build();

            Transition transition = Transition.builder()
                .storageClass(TransitionStorageClass.GLACIER)
                .days(3650)
                .build();

            LifecycleRule rule1 = LifecycleRule.builder()
```

```
                .id("NewRule")
                .filter(ruleFilter)
                .transitions(transition)
                .status(ExpirationStatus.ENABLED)
                .build();

            // Add the new rule to the list.
            newList.add(rule1);
            BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
                .rules(newList)
                .build();

            PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
                .builder()
                .bucket(bucketName)
                .lifecycleConfiguration(lifecycleConfiguration)
                .expectedBucketOwner(accountId)
                .build();


s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Deletes the lifecycle configuration for an Amazon S3 bucket.
     *
     * @param s3 the {@link S3Client} to use for the operation
     * @param bucketName the name of the S3 bucket
     * @param accountId the expected account owner of the S3 bucket
     *
     * @throws S3Exception if an error occurs while deleting the lifecycle
configuration
     */
    public static void deleteLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
        try {
```

```
            DeleteBucketLifecycleRequest deleteBucketLifecycleRequest =
 DeleteBucketLifecycleRequest
                .builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            s3.deleteBucketLifecycle(deleteBucketLifecycleRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[PutBucketLifecycleConfiguration](#)中的。

`PutBucketPolicy`

以下代码示例演示了如何使用 `PutBucketPolicy`。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;

import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
```

```
import java.nio.file.Paths;
import java.util.List;

import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetBucketPolicy {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <polFile>

            Where:
                bucketName - The Amazon S3 bucket to set the policy on.
                polFile - A JSON file containing the policy (see the Amazon S3
 Readme for an example).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String polFile = args[1];
        String policyText = getBucketPolicyFromFile(polFile);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setPolicy(s3, bucketName, policyText);
        s3.close();
    }
```

```
    /**
     * Sets the policy for an Amazon S3 bucket.
     *
     * @param s3         the {@link S3Client} object used to interact with the
Amazon S3 service
     * @param bucketName the name of the Amazon S3 bucket
     * @param policyText the text of the policy to be set on the bucket
     * @throws S3Exception if there is an error setting the bucket policy
     */
    public static void setPolicy(S3Client s3, String bucketName, String policyText)
{
        System.out.println("Setting policy:");
        System.out.println("----");
        System.out.println(policyText);
        System.out.println("----");
        System.out.format("On Amazon S3 bucket: \"%s\"\n", bucketName);

        try {
            PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
                .bucket(bucketName)
                .policy(policyText)
                .build();

            s3.putBucketPolicy(policyReq);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        System.out.println("Done!");
    }

    /**
     * Retrieves the bucket policy from a specified file.
     *
     * @param policyFile the path to the file containing the bucket policy
     * @return the content of the bucket policy file as a string
     */
    public static String getBucketPolicyFromFile(String policyFile) {
        StringBuilder fileText = new StringBuilder();
        try {
            List<String> lines = Files.readAllLines(Paths.get(policyFile),
StandardCharsets.UTF_8);
```

```
            for (String line : lines) {
                fileText.append(line);
            }

        } catch (IOException e) {
            System.out.format("Problem reading file: \"%s\"", policyFile);
            System.out.println(e.getMessage());
        }

        try {
            final JsonParser parser = new
 ObjectMapper().getFactory().createParser(fileText.toString());
            while (parser.nextToken() != null) {
            }

        } catch (IOException jpe) {
            jpe.printStackTrace();
        }
        return fileText.toString();
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutBucketPolicy中的。

`PutBucketReplication`

以下代码示例演示了如何使用 `PutBucketReplication`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Sets the replication configuration for an Amazon S3 bucket.
     *
     * @param s3Client             the S3Client instance to use for the operation
     * @param sourceBucketName     the name of the source bucket
```

```
    * @param destBucketName       the name of the destination bucket
    * @param destinationBucketARN the Amazon Resource Name (ARN) of the destination
bucket
    * @param roleARN              the ARN of the IAM role to use for the
replication configuration
    */
   public static void setReplication(S3Client s3Client, String sourceBucketName,
String destBucketName, String destinationBucketARN, String roleARN) {
       try {
           Destination destination = Destination.builder()
               .bucket(destinationBucketARN)
               .storageClass(StorageClass.STANDARD)
               .build();

           // Define a prefix filter for replication.
           ReplicationRuleFilter ruleFilter = ReplicationRuleFilter.builder()
               .prefix("documents/")
               .build();

           // Define delete marker replication setting.
           DeleteMarkerReplication deleteMarkerReplication =
DeleteMarkerReplication.builder()
               .status(DeleteMarkerReplicationStatus.DISABLED)
               .build();

           // Create the replication rule.
           ReplicationRule replicationRule = ReplicationRule.builder()
               .priority(1)
               .filter(ruleFilter)
               .status(ReplicationRuleStatus.ENABLED)
               .deleteMarkerReplication(deleteMarkerReplication)
               .destination(destination)
               .build();

           List<ReplicationRule> replicationRuleList = new ArrayList<>();
           replicationRuleList.add(replicationRule);

           // Define the replication configuration with IAM role.
           ReplicationConfiguration configuration =
ReplicationConfiguration.builder()
               .role(roleARN)
               .rules(replicationRuleList)
               .build();
```

```
            // Apply the replication configuration to the source bucket.
            PutBucketReplicationRequest replicationRequest =
PutBucketReplicationRequest.builder()
                .bucket(sourceBucketName)
                .replicationConfiguration(configuration)
                .build();

            s3Client.putBucketReplication(replicationRequest);
            System.out.println("Replication configuration set successfully.");

        } catch (IllegalArgumentException e) {
            System.err.println("Configuration error: " + e.getMessage());
        } catch (S3Exception e) {
            System.err.println("S3 Exception: " +
e.awsErrorDetails().errorMessage());
            System.err.println("Status Code: " + e.statusCode());
            System.err.println("Error Code: " + e.awsErrorDetails().errorCode());


        } catch (SdkException e) {
            System.err.println("SDK Exception: " + e.getMessage());
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutBucketReplication中的。

`PutBucketVersioning`

以下代码示例演示了如何使用 `PutBucketVersioning`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * Enables bucket versioning for the specified S3 bucket.
     *
```

```
     * @param s3Client the S3 client to use for the operation
     * @param bucketName the name of the S3 bucket to enable versioning for
     */
    public static void enableBucketVersioning(S3Client s3Client, String bucketName){
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .status(BucketVersioningStatus.ENABLED)
            .build();

        PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        s3Client.putBucketVersioning(versioningRequest);
        System.out.println("Bucket versioning has been enabled for "+bucketName);
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutBucketVersioning中的。

`PutBucketWebsite`

以下代码示例演示了如何使用 `PutBucketWebsite`。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.IndexDocument;
import software.amazon.awssdk.services.s3.model.PutBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.WebsiteConfiguration;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class SetWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = """

            Usage:    <bucketName> [indexdoc]\s

            Where:
               bucketName   - The Amazon S3 bucket to set the website configuration
 on.\s
               indexdoc - The index document, ex. 'index.html'
                          If not specified, 'index.html' will be set.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String indexDoc = "index.html";
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setWebsiteConfig(s3, bucketName, indexDoc);
        s3.close();
    }

    /**
     * Sets the website configuration for an Amazon S3 bucket.
     *
     * @param s3 The {@link S3Client} instance to use for the AWS SDK operations.
     * @param bucketName The name of the S3 bucket to configure.
```

```
     * @param indexDoc The name of the index document to use for the website
 configuration.
     */
    public static void setWebsiteConfig(S3Client s3, String bucketName, String
 indexDoc) {
        try {
            WebsiteConfiguration websiteConfig = WebsiteConfiguration.builder()
                .indexDocument(IndexDocument.builder().suffix(indexDoc).build())
                .build();

            PutBucketWebsiteRequest pubWebsiteReq =
 PutBucketWebsiteRequest.builder()
                .bucket(bucketName)
                .websiteConfiguration(websiteConfig)
                .build();

            s3.putBucketWebsite(pubWebsiteReq);
            System.out.println("The call was successful");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutBucketWebsite中的。

`PutObject`

以下代码示例演示了如何使用 `PutObject`。

适用于 Java 的 SDK 2.x

> **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。

使用 S3Client 将文件上传到存储桶。

```
   /**
    * Uploads a local file to an AWS S3 bucket asynchronously.
    *
    * @param bucketName the name of the S3 bucket to upload the file to
    * @param key        the key (object name) to use for the uploaded file
    * @param objectPath the local file path of the file to be uploaded
    * @return a {@link CompletableFuture} that completes with the {@link
PutObjectResponse} when the upload is successful, or throws a {@link
RuntimeException} if the upload fails
    */
   public CompletableFuture<PutObjectResponse> uploadLocalFileAsync(String
bucketName, String key, String objectPath) {
       PutObjectRequest objectRequest = PutObjectRequest.builder()
           .bucket(bucketName)
           .key(key)
           .build();

       CompletableFuture<PutObjectResponse> response =
getAsyncClient().putObject(objectRequest,
AsyncRequestBody.fromFile(Paths.get(objectPath)));
       return response.whenComplete((resp, ex) -> {
           if (ex != null) {
               throw new RuntimeException("Failed to upload file", ex);
           }
       });
   }
```

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。

```
    /**
     * Puts tags on an Amazon S3 object.
     *
     * @param s3 An {@link S3Client} object that represents the Amazon S3 client.
     * @param bucketName The name of the Amazon S3 bucket.
     * @param objectKey The key of the Amazon S3 object.
     * @param objectPath The file path of the object to be uploaded.
     */
    public static void putS3ObjectTags(S3Client s3, String bucketName, String
objectKey, String objectPath) {
        try {
            Tag tag1 = Tag.builder()
                .key("Tag 1")
                .value("This is tag 1")
                .build();

            Tag tag2 = Tag.builder()
                .key("Tag 2")
                .value("This is tag 2")
                .build();

            List<Tag> tags = new ArrayList<>();
            tags.add(tag1);
            tags.add(tag2);

            Tagging allTags = Tagging.builder()
```

```
                .tagSet(tags)
                .build();

            PutObjectRequest putOb = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .tagging(allTags)
                .build();

            s3.putObject(putOb, RequestBody.fromBytes(getObjectFile(objectPath)));

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Updates the tags associated with an object in an Amazon S3 bucket.
     *
     * @param s3 an instance of the S3Client class, which is used to interact with
the Amazon S3 service
     * @param bucketName the name of the S3 bucket containing the object
     * @param objectKey the key (or name) of the object in the S3 bucket
     * @throws S3Exception if there is an error updating the object's tags
     */
    public static void updateObjectTags(S3Client s3, String bucketName, String
objectKey) {
        try {
            GetObjectTaggingRequest taggingRequest =
GetObjectTaggingRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectTaggingResponse getTaggingRes =
s3.getObjectTagging(taggingRequest);
            List<Tag> obTags = getTaggingRes.tagSet();
            for (Tag sinTag : obTags) {
                System.out.println("The tag key is: " + sinTag.key());
                System.out.println("The tag value is: " + sinTag.value());
            }

            // Replace the object's tags with two new tags.
```

```
            Tag tag3 = Tag.builder()
                .key("Tag 3")
                .value("This is tag 3")
                .build();

            Tag tag4 = Tag.builder()
                .key("Tag 4")
                .value("This is tag 4")
                .build();

            List<Tag> tags = new ArrayList<>();
            tags.add(tag3);
            tags.add(tag4);

            Tagging updatedTags = Tagging.builder()
                .tagSet(tags)
                .build();

            PutObjectTaggingRequest taggingRequest1 =
PutObjectTaggingRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .tagging(updatedTags)
                .build();

            s3.putObjectTagging(taggingRequest1);
            GetObjectTaggingResponse getTaggingRes2 =
s3.getObjectTagging(taggingRequest);
            List<Tag> modTags = getTaggingRes2.tagSet();
            for (Tag sinTag : modTags) {
                System.out.println("The tag key is: " + sinTag.key());
                System.out.println("The tag value is: " + sinTag.value());
            }

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Retrieves the contents of a file as a byte array.
     *
     * @param filePath the path of the file to be read
```

```
     * @return a byte array containing the contents of the file, or null if an error
 occurs
     */
    private static byte[] getObjectFile(String filePath) {
        FileInputStream fileInputStream = null;
        byte[] bytesArray = null;

        try {
            File file = new File(filePath);
            bytesArray = new byte[(int) file.length()];
            fileInputStream = new FileInputStream(file);
            fileInputStream.read(bytesArray);

        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            if (fileInputStream != null) {
                try {
                    fileInputStream.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }

        return bytesArray;
    }
}
```

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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutObjectMetadata {
    public static void main(String[] args) {
        final String USAGE = """

            Usage:
              <bucketName> <objectKey> <objectPath>\s

            Where:
              bucketName - The Amazon S3 bucket to upload an object into.
              objectKey - The object to upload (for example, book.pdf).
              objectPath - The path where the file is located (for example, C:/AWS/
book2.pdf).\s
            """;

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        String objectPath = args[2];
        System.out.println("Putting object " + objectKey + " into bucket " +
 bucketName);
        System.out.println("  in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        putS3Object(s3, bucketName, objectKey, objectPath);
        s3.close();
    }

    /**
     * Uploads an object to an Amazon S3 bucket with metadata.
```

```
     *
     * @param s3 the S3Client object used to interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket to upload the object to
     * @param objectKey the name of the object to be uploaded
     * @param objectPath the local file path of the object to be uploaded
     */
    public static void putS3Object(S3Client s3, String bucketName, String objectKey,
 String objectPath) {
        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("author", "Mary Doe");
            metadata.put("version", "1.0.0.0");

            PutObjectRequest putOb = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .metadata(metadata)
                .build();

            s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
            System.out.println("Successfully placed " + objectKey + " into bucket "
 + bucketName);

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class PutObjectRetention {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <key> <bucketName>\s

            Where:
                key - The name of the object (for example, book.pdf).\s
                bucketName - The Amazon S3 bucket name that contains the object (for
 example, bucket1).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String key = args[0];
        String bucketName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setRentionPeriod(s3, key, bucketName);
        s3.close();
    }

    /**
     * Sets the retention period for an object in an Amazon S3 bucket.
     *
     * @param s3     the S3Client object used to interact with the Amazon S3 service
```

```
     * @param key    the key (name) of the object in the S3 bucket
     * @param bucket the name of the S3 bucket where the object is stored
     *
     * @throws S3Exception if an error occurs while setting the object retention
 period
     */
    public static void setRentionPeriod(S3Client s3, String key, String bucket) {
        try {
            LocalDate localDate = LocalDate.parse("2020-07-17");
            LocalDateTime localDateTime = localDate.atStartOfDay();
            Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

            ObjectLockRetention lockRetention = ObjectLockRetention.builder()
                .mode("COMPLIANCE")
                .retainUntilDate(instant)
                .build();

            PutObjectRetentionRequest retentionRequest =
 PutObjectRetentionRequest.builder()
                .bucket(bucket)
                .key(key)
                .bypassGovernanceRetention(true)
                .retention(lockRetention)
                .build();

            // To set Retention on an object, the Amazon S3 bucket must support
 object
            // locking, otherwise an exception is thrown.
            s3.putObjectRetention(retentionRequest);
            System.out.print("An object retention configuration was successfully
 placed on the object");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutObject中的。

PutObjectLegalHold

以下代码示例演示了如何使用 PutObjectLegalHold。

适用于 Java 的 SDK 2.x

> **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。

```
   // 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutObjectLegalHold中的。

PutObjectLockConfiguration

以下代码示例演示了如何使用 PutObjectLockConfiguration。

适用于 Java 的 SDK 2.x

> **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。

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 rention = DefaultRetention.builder()
            .days(1)
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .build();

        ObjectLockRule lockRule = ObjectLockRule.builder()
            .defaultRetention(rention)
            .build();

        ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
            .objectLockEnabled(ObjectLockEnabled.ENABLED)
            .rule(lockRule)
            .build();

        PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .objectLockConfiguration(objectLockConfiguration)
            .build();
```

```
        getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
        System.out.println("Added a default retention to bucket "+bucketName +".");
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[PutObjectLockConfiguration](中的。

`PutObjectRetention`

以下代码示例演示了如何使用 `PutObjectRetention`。

适用于 Java 的 SDK 2.x

> **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。

```
    // 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutObjectRetention中的。

`RestoreObject`

以下代码示例演示了如何使用 `RestoreObject`。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.RestoreRequest;
import software.amazon.awssdk.services.s3.model.GlacierJobParameters;
import software.amazon.awssdk.services.s3.model.RestoreObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tier;

/*
 *  For more information about restoring an object, see "Restoring an archived
 object" at
 *  https://docs.aws.amazon.com/AmazonS3/latest/userguide/restoring-objects.html
```

```
 *
 *  Before running this Java V2 code example, set up your development environment,
 including your credentials.
 *
 *  For more information, see the following documentation topic:
 *
 *  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RestoreObject {
    public static void main(String[] args) {
        final String usage = """

            Usage:
                <bucketName> <keyName> <expectedBucketOwner>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name of an object with a Storage class value of
 Glacier.\s
                expectedBucketOwner - The account that owns the bucket (you can
 obtain this value from the AWS Management Console).\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String expectedBucketOwner = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        restoreS3Object(s3, bucketName, keyName, expectedBucketOwner);
        s3.close();
    }

    /**
     * Restores an S3 object from the Glacier storage class.
     *
```

```
     * @param s3                    an instance of the {@link S3Client} to be used
 for interacting with Amazon S3
     * @param bucketName            the name of the S3 bucket where the object is
 stored
     * @param keyName               the key (object name) of the S3 object to be
 restored
     * @param expectedBucketOwner   the AWS account ID of the expected bucket owner
     */
    public static void restoreS3Object(S3Client s3, String bucketName, String
 keyName, String expectedBucketOwner) {
        try {
            RestoreRequest restoreRequest = RestoreRequest.builder()
                .days(10)

 .glacierJobParameters(GlacierJobParameters.builder().tier(Tier.STANDARD).build())
                .build();

            RestoreObjectRequest objectRequest = RestoreObjectRequest.builder()
                .expectedBucketOwner(expectedBucketOwner)
                .bucket(bucketName)
                .key(keyName)
                .restoreRequest(restoreRequest)
                .build();

            s3.restoreObject(objectRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考RestoreObject中的。

SelectObjectContent

以下代码示例演示了如何使用 SelectObjectContent。

适用于 Java 的 SDK 2.x

> **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。

以下示例演示了一个使用 JSON 对象的查询。该完整示例还演示了 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 = "amzn-s3-demo-bucket-" + UUID.randomUUID();
    static final S3AsyncClient s3AsyncClient = S3AsyncClient.create();
    static String FILE_CSV = "csv";
    static String FILE_JSON = "json";
    static String URL_CSV = "https://raw.githubusercontent.com/mledoze/countries/
master/dist/countries.csv";
    static String URL_JSON = "https://raw.githubusercontent.com/mledoze/countries/
master/dist/countries.json";

    public static void main(String[] args) {
        SelectObjectContentExample selectObjectContentExample = new
 SelectObjectContentExample();
        try {
            SelectObjectContentExample.setUp();
            selectObjectContentExample.runSelectObjectContentMethodForJSON();
            selectObjectContentExample.runSelectObjectContentMethodForCSV();
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
            System.exit(1);
        } finally {
            SelectObjectContentExample.tearDown();
        }
    }

    EventStreamInfo runSelectObjectContentMethodForJSON() {
        // Set up request parameters.
        final String queryExpression = "select * from s3object[*][*] c where c.area
 < 350000";
        final String fileType = FILE_JSON;

        InputSerialization inputSerialization = InputSerialization.builder()
                .json(JSONInput.builder().type(JSONType.DOCUMENT).build())
                .compressionType(CompressionType.NONE)
                .build();

        OutputSerialization outputSerialization = OutputSerialization.builder()
                .json(JSONOutput.builder().recordDelimiter(null).build())
                .build();

        // Build the SelectObjectContentRequest.
        SelectObjectContentRequest select = SelectObjectContentRequest.builder()
                .bucket(BUCKET_NAME)
```

```
                .key(FILE_JSON)
                .expression(queryExpression)
                .expressionType(ExpressionType.SQL)
                .inputSerialization(inputSerialization)
                .outputSerialization(outputSerialization)
                .build();

        EventStreamInfo eventStreamInfo = new EventStreamInfo();
        // Call the selectObjectContent method with the request and a response
handler.
        // Supply an EventStreamInfo object to the response handler to gather
records and information from the response.
        s3AsyncClient.selectObjectContent(select,
buildResponseHandler(eventStreamInfo)).join();

        // Log out information gathered while processing the response stream.
        long recordCount = eventStreamInfo.getRecords().stream().mapToInt(record ->
                record.split("\n").length
        ).sum();
        logger.info("Total records {}: {}", fileType, recordCount);
        logger.info("Visitor onRecords for fileType {} called {} times", fileType,
eventStreamInfo.getCountOnRecordsCalled());
        logger.info("Visitor onStats for fileType {}, {}", fileType,
eventStreamInfo.getStats());
        logger.info("Visitor onContinuations for fileType {}, {}", fileType,
eventStreamInfo.getCountContinuationEvents());
        return eventStreamInfo;
    }

    static SelectObjectContentResponseHandler buildResponseHandler(EventStreamInfo
eventStreamInfo) {
        // Use a Visitor to process the response stream. This visitor logs
information and gathers details while processing.
        final SelectObjectContentResponseHandler.Visitor visitor =
SelectObjectContentResponseHandler.Visitor.builder()
                .onRecords(r -> {
                    logger.info("Record event received.");
                    eventStreamInfo.addRecord(r.payload().asUtf8String());
                    eventStreamInfo.incrementOnRecordsCalled();
                })
                .onCont(ce -> {
                    logger.info("Continuation event received.");
                    eventStreamInfo.incrementContinuationEvents();
                })
```

```
                .onProgress(pe -> {
                    Progress progress = pe.details();
                    logger.info("Progress event received:\n bytesScanned:
{}\nbytesProcessed: {}\nbytesReturned:{}",
                            progress.bytesScanned(),
                            progress.bytesProcessed(),
                            progress.bytesReturned());
                })
                .onEnd(ee -> logger.info("End event received."))
                .onStats(se -> {
                    logger.info("Stats event received.");
                    eventStreamInfo.addStats(se.details());
                })
                .build();

        // Build the SelectObjectContentResponseHandler with the visitor that
 processes the stream.
        return SelectObjectContentResponseHandler.builder()
                .subscriber(visitor).build();
    }

    // The EventStreamInfo class is used to store information gathered while
 processing the response stream.
    static class EventStreamInfo {
        private final List<String> records = new ArrayList<>();
        private Integer countOnRecordsCalled = 0;
        private Integer countContinuationEvents = 0;
        private Stats stats;

        void incrementOnRecordsCalled() {
            countOnRecordsCalled++;
        }

        void incrementContinuationEvents() {
            countContinuationEvents++;
        }

        void addRecord(String record) {
            records.add(record);
        }

        void addStats(Stats stats) {
            this.stats = stats;
        }
```

```
        public List<String> getRecords() {
            return records;
        }

        public Integer getCountOnRecordsCalled() {
            return countOnRecordsCalled;
        }

        public Integer getCountContinuationEvents() {
            return countContinuationEvents;
        }

        public Stats getStats() {
            return stats;
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SelectObjectContent中的。

UploadPartCopy

以下代码示例演示了如何使用 UploadPartCopy。

适用于 Java 的 SDK 2.x

> **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。

```
   public CompletableFuture<String> performMultiCopy(String toBucket, String
bucketName, String key) {
        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
            .bucket(toBucket)
            .key(key)
            .build();
```

```
        getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
            .thenApply(createMultipartUploadResponse -> {
                String uploadId = createMultipartUploadResponse.uploadId();
                System.out.println("Upload ID: " + uploadId);

                UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
                    .sourceBucket(bucketName)
                    .destinationBucket(toBucket)
                    .sourceKey(key)
                    .destinationKey(key)
                    .uploadId(uploadId)  // Use the valid uploadId.
                    .partNumber(1)  // Ensure the part number is correct.
                    .copySourceRange("bytes=0-1023")  // Adjust range as needed
                    .build();

                return getAsyncClient().uploadPartCopy(uploadPartCopyRequest);
            })
            .thenCompose(uploadPartCopyFuture -> uploadPartCopyFuture)
            .whenComplete((uploadPartCopyResponse, exception) -> {
                if (exception != null) {
                    // Handle any exceptions.
                    logger.error("Error during upload part copy: " +
exception.getMessage());
                } else {
                    // Successfully completed the upload part copy.
                    System.out.println("Upload Part Copy completed successfully.
ETag: " + uploadPartCopyResponse.copyPartResult().eTag());
                }
            });
        return null;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[UploadPartCopy](#)中的。

场景

检查存储桶是否存在

以下代码示例演示如何检查存储桶是否存在。

适用于 Java 的 SDK 2.x

> **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。

您可以使用以下doesBucketExists方法来替代适用于 Java 的 SDK V1 Amazons3Client# doesBucketExist V2（字符串）方法。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.http.HttpStatusCode;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.utils.Validate;

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

    public static void main(String[] args) {
        DoesBucketExist doesBucketExist = new DoesBucketExist();

        final S3Client s3SyncClient = S3Client.builder().build();
        final String bucketName = "amzn-s3-demo-bucket"; // Change to the bucket
 name that you want to check.

        boolean exists = doesBucketExist.doesBucketExist(bucketName, s3SyncClient);
        logger.info("Bucket exists: {}", exists);
    }

    /**
     * Checks if the specified bucket exists. Amazon S3 buckets are named in a
 global namespace; use this method to
     * determine if a specified bucket name already exists, and therefore can't be
 used to create a new bucket.
     * <p>
     * Internally this method uses the <a
```

```
     * href="https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/
services/s3/
S3Client.html#getBucketAcl(java.util.function.Consumer)">S3Client.getBucketAcl(String)</
a>
     * operation to determine whether the bucket exists.
     * <p>
     * This method is equivalent to the AWS SDK for Java V1's <a
     * href="https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/
com/amazonaws/services/s3/AmazonS3Client.html#doesBucketExistV2-
java.lang.String-">AmazonS3Client#doesBucketExistV2(String)</a>.
     *
     * @param bucketName   The name of the bucket to check.
     * @param s3SyncClient An <code>S3Client</code> instance. The method checks for
 the bucket in the AWS Region
     *                     configured on the instance.
     * @return The value true if the specified bucket exists in Amazon S3; the value
 false if there is no bucket in
     *         Amazon S3 with that name.
     */
    public boolean doesBucketExist(String bucketName, S3Client s3SyncClient) {
        try {
            Validate.notEmpty(bucketName, "The bucket name must not be null or an
 empty string.", "");
            s3SyncClient.getBucketAcl(r -> r.bucket(bucketName));
            return true;
        } catch (AwsServiceException ase) {
            // A redirect error or an AccessDenied exception means the bucket exists
 but it's not in this region
            // or we don't have permissions to it.
            if ((ase.statusCode() == HttpStatusCode.MOVED_PERMANENTLY) ||
 "AccessDenied".equals(ase.awsErrorDetails().errorCode())) {
                return true;
            }
            if (ase.statusCode() == HttpStatusCode.NOT_FOUND) {
                return false;
            }
            throw ase;
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetBucketAcl中的。

创建预签名 URL

以下代码示例演示了如何为 Amazon S3 创建预签名 URL 以及如何上传对象。

适用于 Java 的 SDK 2.x

> **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。

以下显示了如何创建预签名 URLs 并 URLs 与 HTTP 客户端库一起使用的三个示例：

- 一个 HTTP GET 请求，它将此 URL 与三个 HTTP 客户端库结合使用
- 一个在标头中包含元数据的 HTTP PUT 请求，它将此 URL 与三个 HTTP 客户端库结合使用
- 一个带有查询参数的 HTTP PUT 请求，它将此 URL 与一个 HTTP 客户端库结合使用

为对象生成预签名 URL，然后下载该 URL（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 JDK HttpURLConnection（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 JDK `HttpClient`（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 适用于 Java 的 SDK SdkHttpClient 类进行下载。

```
    /* Use the AWS SDK for Java SdkHttpClient class to do the download. */
    public byte[] useSdkHttpClientToGet(String presignedUrlString) {

        ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.
        try {
            URL presignedUrl = new URL(presignedUrlString);
            SdkHttpRequest request = SdkHttpRequest.builder()
                    .method(SdkHttpMethod.GET)
                    .uri(presignedUrl.toURI())
                    .build();

            HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
                    .request(request)
                    .build();

            try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
                HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
                response.responseBody().ifPresentOrElse(
                        abortableInputStream -> {
                            try {
                                IoUtils.copy(abortableInputStream,
byteArrayOutputStream);
                            } catch (IOException e) {
                                throw new RuntimeException(e);
                            }
                        },
                        () -> logger.error("No response body."));

                logger.info("HTTP Response code is {}",
response.httpResponse().statusCode());
            }
```

```
        } catch (URISyntaxException | IOException e) {
            logger.error(e.getMessage(), e);
        }
        return byteArrayOutputStream.toByteArray();
    }
```

为进行上传生成在标头中带有元数据的预签名 URL，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(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();
       }
   }
```

使用以下三种方法中的任何一种上传文件对象。

使用 JDK HttpURLConnection（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 JDK `HttpClient`（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);
        }
    }
```

为进行上传生成带有查询参数的预签名 URL，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.awscore.AwsRequestOverrideConfiguration;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

import java.io.File;
import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

生成 URL。

```
   /**
    *  Creates a presigned URL to use in a subsequent HTTP PUT request. The code
adds query parameters
    *  to the request instead of using headers. By using query parameters, you do
not need to add the
    *  the parameters as headers when the PUT request is eventually sent.
    *
    * @param bucketName Bucket name where the object will be uploaded.
    * @param keyName Key name of the object that will be uploaded.
    * @param queryParams Query string parameters to be added to the presigned URL.
    * @return
    */
   public String createPresignedUrl(String bucketName, String keyName, Map<String,
String> queryParams) {
       try (S3Presigner presigner = S3Presigner.create()) {
           // Create an override configuration to store the query parameters.
           AwsRequestOverrideConfiguration.Builder overrideConfigurationBuilder =
AwsRequestOverrideConfiguration.builder();

           queryParams.forEach(overrideConfigurationBuilder::putRawQueryParameter);

           PutObjectRequest objectRequest = PutObjectRequest.builder()
                   .bucket(bucketName)
                   .key(keyName)
                   .overrideConfiguration(overrideConfigurationBuilder.build()) //
Add the override configuration.
                   .build();

           PutObjectPresignRequest presignRequest =
PutObjectPresignRequest.builder()
                   .signatureDuration(Duration.ofMinutes(10))  // The URL expires
in 10 minutes.
                   .putObjectRequest(objectRequest)
                   .build();


           PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);
           String myURL = presignedRequest.url().toString();
           logger.info("Presigned URL to upload a file to: [{}]", myURL);
           logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());
```

```
            return presignedRequest.url().toExternalForm();
        }
    }
```

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 execute the PUT request.
Since the
     * URL contains the query parameters, no headers are needed for metadata, SSE
settings, or ACL settings.
     *
     * @param presignedUrlString The URL for the PUT request.
     * @param fileToPut File to uplaod
     */
    public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut) {
        logger.info("Begin [{}] upload", fileToPut.toString());

        try {
            URL presignedUrl = new URL(presignedUrlString);

            SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
                    .method(SdkHttpMethod.PUT)
                    .uri(presignedUrl.toURI());

            SdkHttpRequest request = requestBuilder.build();

            HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
                    .request(request)
                    .contentStreamProvider(new
FileContentStreamProvider(fileToPut.toPath()))
                    .build();

            try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
                HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
                logger.info("Response code: {}",
response.httpResponse().statusCode());
            }
        } catch (URISyntaxException | IOException e) {
            logger.error(e.getMessage(), e);
```

```
        }
    }
```

创建无服务器应用程序来管理照片

以下代码示例演示如何创建无服务器应用程序，让用户能够使用标签管理照片。

适用于 Java 的 SDK 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 Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

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。

适用于 Java 的 SDK 2.x

> **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。

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。

```
   /**
    * Aborts all incomplete multipart uploads from the specified S3 bucket.
    * <p>
    * This method retrieves a list of all incomplete multipart uploads in the
specified S3 bucket,
    * and then aborts each of those uploads.
    */
   public static void abortIncompleteMultipartUploadsFromList() {
       ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
           .bucket(bucketName)
           .build();

       ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
       List<MultipartUpload> uploads = response.uploads();

       AbortMultipartUploadRequest abortMultipartUploadRequest;
       for (MultipartUpload upload : uploads) {
           abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
               .bucket(bucketName)
               .key(upload.key())
               .expectedBucketOwner(accountId)
               .uploadId(upload.uploadId())
               .build();

           AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
           if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
               logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
upload.uploadId(), bucketName);
           }
       }
   }
```

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《AWS SDK for Java 2.x API Reference》中的以下主题。
  - AbortMultipartUpload
  - ListMultipartUploads
  - PutBucketLifecycleConfiguration

检测图像中的 PPE

以下代码示例展示如何构建采用 Amazon Rekognition 来检测图像中的个人防护设备（PPE）的应用程序。

适用于 Java 的 SDK 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
- Amazon SES

检测图像中的对象

以下代码示例演示如何构建采用 Amazon Rekognition 来按类别检测图像中物体的应用程序。

适用于 Java 的 SDK 2.x

展示如何使用 Amazon Rekognition Java API 创建应用程序，该应用程序采用 Amazon Rekognition 来按类别识别位于 Amazon Simple Storage Service (Amazon S3) 存储桶的图像当中的对象。该应用程序使用 Amazon Simple Email Service (Amazon SES) 向管理员发送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
- Amazon SES

检测视频中的人物和对象

以下代码示例演示如何使用 Amazon Rekognition 检测视频中的人物和物体。

适用于 Java 的 SDK 2.x

展示如何使用 Amazon Rekognition Java API 创建应用程序，以检测位于 Amazon Simple Storage Service (Amazon S3) 存储桶的视频当中的人脸和对象。该应用程序使用 Amazon Simple Email Service (Amazon SES) 向管理员发送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
- Amazon SES
- Amazon SNS
- Amazon SQS

下载 S3“目录”

以下代码示例演示如何下载和筛选 Amazon S3 存储桶“目录”的内容。

适用于 Java 的 SDK 2.x

> **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。

此示例说明如何使用TransferManager中的 S3 从 Amazon S3 存储桶下载 “目录”。 AWS SDK for Java 2.x 它还演示了如何在请求DownloadFilters中使用。

```
    /**
     * For standard buckets, S3 provides the illusion of a directory structure
through the use of keys. When you upload
     * an object to an S3 bucket, you specify a key, which is essentially the "path"
to the object. The key can contain
     * forward slashes ("/") to make it appear as if the object is stored in a
directory structure, but this is just a
     * logical representation, not an actual directory.
     * <p><pre>
     * In this example, our S3 bucket contains the following objects:
     *
     * folder1/file1.txt
     * folder1/file2.txt
     * folder1/file3.txt
     * folder2/file1.txt
     * folder2/file2.txt
```

```
    * folder2/file3.txt
    * folder3/file1.txt
    * folder3/file2.txt
    * folder3/file3.txt
    *
    * When method `downloadS3Directories` is invoked with
    * `destinationPathURI` set to `/test`, the downloaded
    * directory looks like:
    *
    * |- test
    *    |- folder1
    *       |- file1.txt
    *       |- file2.txt
    *       |- file3.txt
    *    |- folder3
    *       |- file1.txt
    *       |- file2.txt
    *       |- file3.txt
    * </pre>
    *
    * @param transferManager    An S3TransferManager instance.
    * @param destinationPathURI local directory to hold the downloaded S3
'directories' and files.
    * @param bucketName         The S3 bucket that contains the 'directories' to
download.
    * @return The number of objects (files, in this case) that were downloaded.
    */
   public Integer downloadS3Directories(S3TransferManager transferManager,
                                        URI destinationPathURI, String bucketName)
{

       // Define the filters for which 'directories' we want to download.
       DownloadFilter folder1Filter = (S3Object s3Object) ->
s3Object.key().startsWith("folder1/");
       DownloadFilter folder3Filter = (S3Object s3Object) ->
s3Object.key().startsWith("folder3/");
       DownloadFilter folderFilter = s3Object ->
folder1Filter.or(folder3Filter).test(s3Object);

       DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
               .destination(Paths.get(destinationPathURI))
               .bucket(bucketName)
               .filter(folderFilter)
```

```
                .build());
        CompletedDirectoryDownload completedDirectoryDownload =
 directoryDownload.completionFuture().join();

        Integer numFilesInFolder1 =
 Paths.get(destinationPathURI).resolve("folder1").toFile().list().length;
        Integer numFilesInFolder3 =
 Paths.get(destinationPathURI).resolve("folder3").toFile().list().length;

        try {
            assert numFilesInFolder1 == 3;
            assert numFilesInFolder3 == 3;
            assert !
Paths.get(destinationPathURI).resolve("folder2").toFile().exists(); // `folder2` was
 not downloaded.
        } catch (AssertionError e) {
            logger.error("An assertion failed.");
        }

        completedDirectoryDownload.failedTransfers()
                .forEach(fail -> logger.warn("Object failed to transfer  [{}]",
 fail.exception().getMessage()));
        return numFilesInFolder1 + numFilesInFolder3;
    }
```

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。

适用于 Java 的 SDK 2.x

> **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。

使用 S 3 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DownloadDirectory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。

适用于 Java 的 SDK 2.x

> **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。

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.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.stream.Collectors;

/*
 Before running this Java V2 code example, set up your development
 environment, including your credentials.

 For more information, see the following documentation topic:
 https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html

 This Java example performs the following tasks:
    1. Create test Amazon Simple Storage Service (S3) buckets with different lock
 policies.
    2. Upload sample objects to each bucket.
    3. Set some Legal Hold and Retention Periods on objects and buckets.
    4. Investigate lock policies by viewing settings or attempting to delete or
 overwrite objects.
    5. Clean up objects and buckets.
 */
public class S3ObjectLockWorkflow {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    static String bucketName;
    static S3LockActions s3LockActions;
    private static final List<String> bucketNames = new ArrayList<>();
    private static final List<String> fileNames = new ArrayList<>();
```

```
    public static void main(String[] args) {
        final String usage = """
            Usage:
                <bucketName> \s

            Where:
                bucketName - The Amazon S3 bucket name.
           """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
        s3LockActions = new S3LockActions();
        bucketName = args[0];
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Simple Storage Service (S3) Object
Locking Feature Scenario.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        configurationSetup();
        System.out.println(DASHES);

        System.out.println(DASHES);
        setup();
        System.out.println("Setup is complete. Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Lets present the user with choices.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        demoActionChoices() ;
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Would you like to clean up the resources? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            cleanup();
            System.out.println("Clean up is complete.");
```

```
        }

        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Amazon S3 Object Locking Workflow is complete.");
        System.out.println(DASHES);
    }

    // Present the user with the demo action choices.
    public static void demoActionChoices() {
        String[] choices = {
            "List all files in buckets.",
            "Attempt to delete a file.",
            "Attempt to delete a file with retention period bypass.",
            "Attempt to overwrite a file.",
            "View the object and bucket retention settings for a file.",
            "View the legal hold settings for a file.",
            "Finish the workflow."
        };

        int choice = 0;
        while (true) {
            System.out.println(DASHES);
            choice = getChoiceResponse("Explore the S3 locking features by selecting
one of the following choices:", choices);
            System.out.println(DASHES);
            System.out.println("You selected "+choices[choice]);
            switch (choice) {
                case 0 -> {
                    s3LockActions.listBucketsAndObjects(bucketNames, true);
                }

                case 1 -> {
                    System.out.println("Enter the number of the object to delete:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
```

```
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    String version = allFiles.get(fileChoice).getVersion();
                    s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
false, version);
                }

                case 2 -> {
                    System.out.println("Enter the number of the object to delete:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    String version = allFiles.get(fileChoice).getVersion();
                    s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
true, version);
                }

                case 3 -> {
                    System.out.println("Enter the number of the object to
overwrite:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();

                    // Attempt to overwrite the file.
                    try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(objectKey))) {
                        writer.write("This is a modified text.");

                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                    s3LockActions.uploadFile(bucketName, objectKey, objectKey);
                }
```

```
                case 4 -> {
                    System.out.println("Enter the number of the object to
overwrite:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    s3LockActions.getObjectRetention(bucketName, objectKey);
                }

                case 5 -> {
                    System.out.println("Enter the number of the object to view:");
                    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                    List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                    String[] fileKeysArray = fileKeys.toArray(new String[0]);
                    int fileChoice = getChoiceResponse(null, fileKeysArray);
                    String objectKey = fileKeys.get(fileChoice);
                    String bucketName = allFiles.get(fileChoice).getBucketName();
                    s3LockActions.getObjectLegalHold(bucketName, objectKey);
                    s3LockActions.getBucketObjectLockConfiguration(bucketName);
                }

                case 6 -> {
                    System.out.println("Exiting the workflow...");
                    return;
                }

                default -> {
                    System.out.println("Invalid choice. Please select again.");
                }
            }
        }
    }

    // Clean up the resources from the scenario.
    private static void cleanup() {
        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, false);
```

```
        for (S3InfoObject fileInfo : allFiles) {
            String bucketName = fileInfo.getBucketName();
            String key = fileInfo.getKeyName();
            String version = fileInfo.getVersion();
            if (bucketName.contains("lock-enabled") ||
(bucketName.contains("retention-after-creation"))) {
                ObjectLockLegalHold legalHold =
s3LockActions.getObjectLegalHold(bucketName, key);
                if (legalHold != null) {
                    String holdStatus = legalHold.status().name();
                    System.out.println(holdStatus);
                    if (holdStatus.compareTo("ON") == 0) {
                        s3LockActions.modifyObjectLegalHold(bucketName, key, false);
                    }
                }
                // Check for a retention period.
                ObjectLockRetention retention =
s3LockActions.getObjectRetention(bucketName, key);
                boolean hasRetentionPeriod ;
                hasRetentionPeriod = retention != null;
                s3LockActions.deleteObjectFromBucket(bucketName,
key,hasRetentionPeriod, version);

            } else {
                System.out.println(bucketName +" objects do not have a legal lock");
                s3LockActions.deleteObjectFromBucket(bucketName, key,false,
version);
            }
        }

        // Delete the buckets.
        System.out.println("Delete "+bucketName);
        for (String bucket : bucketNames){
            s3LockActions.deleteBucketByName(bucket);
        }
    }

    private static void setup() {
        Scanner scanner = new Scanner(System.in);
        System.out.println("""
                For this workflow, we will use the AWS SDK for Java to create
several S3
                buckets and files to demonstrate working with S3 locking features.
                """);
```

```
        System.out.println("S3 buckets can be created either with or without object
lock enabled.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();

        // Create three S3 buckets.
        s3LockActions.createBucketWithLockOptions(false, bucketNames.get(0));
        s3LockActions.createBucketWithLockOptions(true, bucketNames.get(1));
        s3LockActions.createBucketWithLockOptions(false, bucketNames.get(2));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();

        System.out.println("Bucket "+bucketNames.get(2) +" will be configured to use
object locking with a default retention period.");
        s3LockActions.modifyBucketDefaultRetention(bucketNames.get(2));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();

        System.out.println("Object lock policies can also be added to existing
buckets. For this example, we will use "+bucketNames.get(1));
        s3LockActions.enableObjectLockOnBucket(bucketNames.get(1));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();

        // Upload some files to the buckets.
        System.out.println("Now let's add some test files:");
        String fileName = "exampleFile.txt";
        int fileCount = 2;
        try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(fileName))) {
            writer.write("This is a sample file for uploading to a bucket.");

        } catch (IOException e) {
            e.printStackTrace();
        }

        for (String bucketName : bucketNames){
            for (int i = 0; i < fileCount; i++) {
                // Get the file name without extension.
                String fileNameWithoutExtension =
java.nio.file.Paths.get(fileName).getFileName().toString();
                int extensionIndex = fileNameWithoutExtension.lastIndexOf('.');
                if (extensionIndex > 0) {
```

```
                    fileNameWithoutExtension = fileNameWithoutExtension.substring(0,
extensionIndex);
                }

                // Create the numbered file names.
                String numberedFileName = fileNameWithoutExtension + i +
getFileExtension(fileName);
                fileNames.add(numberedFileName);
                s3LockActions.uploadFile(bucketName, numberedFileName, fileName);
            }
        }

        String question = null;
        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        System.out.println("Now we can set some object lock policies on individual
files:");
        for (String bucketName : bucketNames) {
            for (int i = 0; i < fileNames.size(); i++){

                // No modifications to the objects in the first bucket.
                if (!bucketName.equals(bucketNames.get(0))) {
                    String exampleFileName = fileNames.get(i);
                    switch (i) {
                        case 0 -> {
                            question = "Would you like to add a legal hold to " +
exampleFileName + " in " + bucketName + " (y/n)?";
                            System.out.println(question);
                            String ans = scanner.nextLine().trim();
                            if (ans.equalsIgnoreCase("y")) {
                                System.out.println("**** You have selected to put a
legal hold " + exampleFileName);

                                // Set a legal hold.
                                s3LockActions.modifyObjectLegalHold(bucketName,
exampleFileName, true);
                            }
                        }
                        case 1 -> {
                            """
                                Would you like to add a 1 day Governance retention
period to %s in %s (y/n)?
```

```
                                Reminder: Only a user with the
 s3:BypassGovernanceRetention permission will be able to delete this file or its
 bucket until the retention period has expired.
                                """.formatted(exampleFileName, bucketName);
                            System.out.println(question);
                            String ans2 = scanner.nextLine().trim();
                            if (ans2.equalsIgnoreCase("y")) {

 s3LockActions.modifyObjectRetentionPeriod(bucketName, exampleFileName);
                            }
                        }
                    }
                }
            }
        }

    // Get file extension.
    private static String getFileExtension(String fileName) {
        int dotIndex = fileName.lastIndexOf('.');
        if (dotIndex > 0) {
            return fileName.substring(dotIndex);
        }
        return "";
    }

    public static void configurationSetup() {
        String noLockBucketName = bucketName + "-no-lock";
        String lockEnabledBucketName = bucketName + "-lock-enabled";
        String retentionAfterCreationBucketName = bucketName + "-retention-after-
creation";
        bucketNames.add(noLockBucketName);
        bucketNames.add(lockEnabledBucketName);
        bucketNames.add(retentionAfterCreationBucketName);
    }

    public static int getChoiceResponse(String question, String[] choices) {
        Scanner scanner = new Scanner(System.in);
        if (question != null) {
            System.out.println(question);
            for (int i = 0; i < choices.length; i++) {
                System.out.println("\t" + (i + 1) + ". " + choices[i]);
            }
        }
```

```
        int choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.length) {
            String choice = scanner.nextLine();
            try {
                choiceNumber = Integer.parseInt(choice);
            } catch (NumberFormatException e) {
                System.out.println("Invalid choice. Please enter a valid number.");
            }
        }

        return choiceNumber - 1;
    }
}
```

S3 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 rention = DefaultRetention.builder()
            .days(1)
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .build();

        ObjectLockRule lockRule = ObjectLockRule.builder()
            .defaultRetention(rention)
            .build();

        ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
            .objectLockEnabled(ObjectLockEnabled.ENABLED)
            .rule(lockRule)
            .build();

        PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
```

```
            .bucket(bucketName)
            .objectLockConfiguration(objectLockConfiguration)
            .build();

        getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
        System.out.println("Added a default retention to bucket "+bucketName +".");
    }

    // Enable object lock on an existing bucket.
    public void enableObjectLockOnBucket(String bucketName) {
        try {
            VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
                .status(BucketVersioningStatus.ENABLED)
                .build();

            PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
                .bucket(bucketName)
                .versioningConfiguration(versioningConfiguration)
                .build();

            // Enable versioning on the bucket.
            getClient().putBucketVersioning(putBucketVersioningRequest);
            PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
                .bucket(bucketName)
                .objectLockConfiguration(ObjectLockConfiguration.builder()
                    .objectLockEnabled(ObjectLockEnabled.ENABLED)
                    .build())
                .build();

            getClient().putObjectLockConfiguration(request);
            System.out.println("Successfully enabled object lock on "+bucketName);

        } catch (S3Exception ex) {
            System.out.println("Error modifying object lock: '" + ex.getMessage() +
"'");
        }
    }

    public void uploadFile(String bucketName, String objectName, String filePath) {
        Path file = Paths.get(filePath);
        PutObjectRequest request = PutObjectRequest.builder()
```

```
            .bucket(bucketName)
            .key(objectName)
            .checksumAlgorithm(ChecksumAlgorithm.SHA256)
            .build();

        PutObjectResponse response = getClient().putObject(request, file);
        if (response != null) {
            System.out.println("\tSuccessfully uploaded " + objectName + " to " +
bucketName + ".");
        } else {
            System.out.println("\tCould not upload " + objectName + " to " +
bucketName + ".");
        }
    }

    // Set or modify a legal hold on an object in an S3 bucket.
    public void modifyObjectLegalHold(String bucketName, String objectKey, boolean
legalHoldOn) {
        ObjectLockLegalHold legalHold ;
        if (legalHoldOn) {
            legalHold = ObjectLockLegalHold.builder()
                .status(ObjectLockLegalHoldStatus.ON)
                .build();
        } else {
            legalHold = ObjectLockLegalHold.builder()
                .status(ObjectLockLegalHoldStatus.OFF)
                .build();
        }

        PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .legalHold(legalHold)
            .build();

        getClient().putObjectLegalHold(legalHoldRequest) ;
        System.out.println("Modified legal hold for "+ objectKey +" in "+bucketName
+".");
    }

    // Delete an object from a specific bucket.
    public void deleteObjectFromBucket(String bucketName, String objectKey, boolean
hasRetention, String versionId) {
```

```
        try {
            DeleteObjectRequest objectRequest;
            if (hasRetention) {
                objectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .versionId(versionId)
                    .bypassGovernanceRetention(true)
                    .build();
            } else {
                objectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .versionId(versionId)
                    .build();
            }

            getClient().deleteObject(objectRequest) ;
            System.out.println("The object was successfully deleted");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Get the retention period for an S3 object.
    public ObjectLockRetention getObjectRetention(String bucketName, String key){
        try {
            GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

            GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
            System.out.println("tObject retention for "+key +" in "+ bucketName +":
" + response.retention().mode() +" until "+ response.retention().retainUntilDate()
+".");
            return response.retention();

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            return null;
```

```
        }
    }

    public void deleteBucketByName(String bucketName) {
        try {
            DeleteBucketRequest request = DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();

            getClient().deleteBucket(request);
            System.out.println(bucketName +" was deleted.");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }

    // Get the object lock configuration details for an S3 bucket.
    public void getBucketObjectLockConfiguration(String bucketName) {
        GetObjectLockConfigurationRequest objectLockConfigurationRequest =
 GetObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .build();

        GetObjectLockConfigurationResponse response =
 getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
        System.out.println("Bucket object lock config for "+bucketName +":  ");
        System.out.println("\tEnabled:
 "+response.objectLockConfiguration().objectLockEnabled());
        System.out.println("\tRule: "+
 response.objectLockConfiguration().rule().defaultRetention());
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold
  - PutObjectLockConfiguration

- PutObjectRetention

使用 S3 管理大型消息

以下代码示例演示如何使用 Amazon SQS 扩展型客户端库来处理大型 Amazon SQS 消息。

适用于 Java 的 SDK 2.x

> **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。

```
import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.joda.time.DateTime;
import org.joda.time.format.DateTimeFormat;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.LifecycleExpiration;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueResponse;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
```

```
import software.amazon.awssdk.services.sqs.model.ReceiveMessageResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;

import java.util.Arrays;
import java.util.List;
import java.util.UUID;

/**
 * Example of using Amazon SQS Extended Client Library for Java 2.x.
 */
public class SqsExtendedClientExample {
    private static final Logger logger =
 LoggerFactory.getLogger(SqsExtendedClientExample.class);

    private String s3BucketName;
    private String queueUrl;
    private final String queueName;
    private final S3Client s3Client;
    private final SqsClient sqsExtendedClient;
    private final int messageSize;

    /**
     * Constructor with default clients and message size.
     */
    public SqsExtendedClientExample() {
        this(S3Client.create(), 300000);
    }

    /**
     * Constructor with custom S3 client and message size.
     *
     * @param s3Client The S3 client to use
     * @param messageSize The size of the test message to create
     */
    public SqsExtendedClientExample(S3Client s3Client, int messageSize) {
        this.s3Client = s3Client;
        this.messageSize = messageSize;

        // Generate a unique bucket name.
        this.s3BucketName = UUID.randomUUID() + "-" +
                DateTimeFormat.forPattern("yyMMdd-hhmmss").print(new DateTime());

        // Generate a unique queue name.
        this.queueName = "MyQueue-" + UUID.randomUUID();
```

```
        // Configure the SQS extended client.
        final ExtendedClientConfiguration extendedClientConfig = new
ExtendedClientConfiguration()
                .withPayloadSupportEnabled(s3Client, s3BucketName);

        this.sqsExtendedClient = new
AmazonSQSExtendedClient(SqsClient.builder().build(), extendedClientConfig);
    }

    public static void main(String[] args) {
        SqsExtendedClientExample example = new SqsExtendedClientExample();
        try {
            example.setup();
            example.sendAndReceiveMessage();
        } finally {
            example.cleanup();
        }
    }

    /**
     * Send a large message and receive it back.
     *
     * @return The received message
     */
    public Message sendAndReceiveMessage() {
        try {
            // Create a large message.
            char[] chars = new char[messageSize];
            Arrays.fill(chars, 'x');
            String largeMessage = new String(chars);

            // Send the message.
            final SendMessageRequest sendMessageRequest =
SendMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageBody(largeMessage)
                    .build();

            sqsExtendedClient.sendMessage(sendMessageRequest);
            logger.info("Sent message of size: {}", largeMessage.length());

            // Receive and return the message.
```

```
            final ReceiveMessageResponse receiveMessageResponse =
sqsExtendedClient.receiveMessage(
                    ReceiveMessageRequest.builder().queueUrl(queueUrl).build());

            List<Message> messages = receiveMessageResponse.messages();
            if (messages.isEmpty()) {
                throw new RuntimeException("No messages received");
            }

            Message message = messages.getFirst();
            logger.info("\nMessage received.");
            logger.info("  ID: {}", message.messageId());
            logger.info("  Receipt handle: {}", message.receiptHandle());
            logger.info("  Message body size: {}", message.body().length());
            logger.info("  Message body (first 5 characters): {}",
message.body().substring(0, 5));

            return message;
        } catch (RuntimeException e) {
            logger.error("Error during message processing: {}", e.getMessage(), e);
            throw e;
        }
    }
```

- 有关更多信息，请参阅《AWS SDK for Java 2.x 开发人员指南》https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-s3-messages.html。
- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - CreateBucket
  - PutBucketLifecycleConfiguration
  - ReceiveMessage
  - SendMessage

解析 URIs

以下代码示例演示如何解析 Amazon S3 URIs 以提取存储桶名称和对象密钥等重要组件。

适用于 Java 的 SDK 2.x

> **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。

使用 S3Uri 类解析 Amazon S3 URI。

```
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);
        }
    }
```

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。

适用于 Java 的 SDK 2.x

> **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。

此示例显示了如何使用 Amazon SQS 处理 S3 通知事件。

```
    /**
     * 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《AWS SDK for Java 2.x API Reference》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允许存储桶向 Amazon SNS 主题和 Amazon SQS 队列发送 S3 事件通知，EventBridge 以及如何将通知路由到 Amazon SNS 主题和 Amazon SQS 队列。

适用于 Java 的 SDK 2.x

> **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。

```
   /** 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《AWS SDK for Java 2.x API Reference》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。

适用于 Java 的 SDK 2.x

> **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。

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 downloadedFileWithPath) {
       DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
               .getObjectRequest(b -> b.bucket(bucketName).key(key))
               .addTransferListener(LoggingTransferListener.create())  // Add
listener.
               .destination(Paths.get(downloadedFileWithPath))
               .build();

       FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

       CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
       /*
```

```
            The SDK provides a LoggingTransferListener implementation of the
TransferListener interface.
            You can also implement the interface to provide your own logic.

            Configure log4J2 with settings such as the following.
                <Configuration status="WARN">
                    <Appenders>
                        <Console name="AlignedConsoleAppender" target="SYSTEM_OUT">
                            <PatternLayout pattern="%m%n"/>
                        </Console>
                    </Appenders>

                    <Loggers>
                        <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
                            <AppenderRef ref="AlignedConsoleAppender"/>
                        </logger>
                    </Loggers>
                </Configuration>

            Log4J2 logs the progress. The following is example output for a 21.3 MB
file download.
                Transfer initiated...
                |=======             | 39.4%
                |===============     | 78.8%
                |====================| 100.0%
                Transfer complete!
        */
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》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。

适用于 Java 的 SDK 2.x

> **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。

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 的详细信息，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。

适用于 Java 的 SDK 2.x

> **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。

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();
    }
```

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;
```

在基于AWS CRT 的 S3 客户端上使用 S3 Transfer Manager，以在内容大小超过阈值时透明地执行分段上传。默认阈值大小为 8 MB。

```
    /**
     * Uploads a file to an Amazon S3 bucket using the S3TransferManager.
     *
     * @param filePath the file path of the file to be uploaded
     */
    public void multipartUploadWithTransferManager(String filePath) {
        S3TransferManager transferManager = S3TransferManager.create();
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
            .putObjectRequest(b -> b
                .bucket(bucketName)
                .key(key))
            .source(Paths.get(filePath))
            .build();
        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
        fileUpload.completionFuture().join();
        transferManager.close();
    }
```

使用 S3Client API 执行分段上传。

```
    /**
     * Performs a multipart upload to Amazon S3 using the provided S3 client.
     *
     * @param filePath the path to the file to be uploaded
```

```
     */
    public void multipartUploadWithS3Client(String filePath) {

        // Initiate the multipart upload.
        CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
            .bucket(bucketName)
            .key(key));
        String uploadId = createMultipartUploadResponse.uploadId();

        // Upload the parts of the file.
        int partNumber = 1;
        List<CompletedPart> completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
            long position = 0;
            while (position < fileSize) {
                file.seek(position);
                long read = file.getChannel().read(bb);

                bb.flip(); // Swap position and limit before reading from the
buffer.
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                    .bucket(bucketName)
                    .key(key)
                    .uploadId(uploadId)
                    .partNumber(partNumber)
                    .build();

                UploadPartResponse partResponse = s3Client.uploadPart(
                    uploadPartRequest,
                    RequestBody.fromByteBuffer(bb));

                CompletedPart part = CompletedPart.builder()
                    .partNumber(partNumber)
                    .eTag(partResponse.eTag())
                    .build();
                completedParts.add(part);

                bb.clear();
                position += read;
                partNumber++;
```

```
            }
        } catch (IOException e) {
            logger.error(e.getMessage());
        }

        // Complete the multipart upload.
        s3Client.completeMultipartUpload(b -> b
            .bucket(bucketName)
            .key(key)
            .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
    }
```

使用启用了多部分支持AsyncClient 的 S3 API 来执行分段上传。

```
    /**
     * Uploads a file to an S3 bucket using the S3AsyncClient and enabling multipart
support.
     *
     * @param filePath the local file path of the file to be uploaded
     */
    public void multipartUploadWithS3AsyncClient(String filePath) {
        // Enable multipart support.
        S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
            .multipartEnabled(true)
            .build();

        CompletableFuture<PutObjectResponse> response = s3AsyncClient.putObject(b ->
b
                .bucket(bucketName)
                .key(key),
            Paths.get(filePath));

        response.join();
        logger.info("File uploaded in multiple 8 MiB parts using S3AsyncClient.");
    }
```

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。

适用于 Java 的 SDK 2.x

> **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。

使用 AWS 基于 CRT 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.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

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

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket-" + UUID.randomUUID(); // Change
 bucket name.
        String key = UUID.randomUUID().toString();

        AsyncExampleUtils.createBucket(bucketName);
        try {
            PutObjectFromStreamAsync example = new PutObjectFromStreamAsync();
            S3AsyncClient s3AsyncClientCrt = S3AsyncClient.crtCreate();
```

```
            PutObjectResponse putObjectResponse =
example.putObjectFromStreamCrt(s3AsyncClientCrt, bucketName, key);
            logger.info("Object {} etag: {}", key, putObjectResponse.eTag());
            logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
        } finally {
            AsyncExampleUtils.deleteObject(bucketName, key);
            AsyncExampleUtils.deleteBucket(bucketName);
        }
    }

    /**
     * @param s33CrtAsyncClient - To upload content from a stream of unknown size,
use can the AWS CRT-based S3 client.
     * @param bucketName - The name of the bucket.
     * @param key - The name of the object.
     * @return software.amazon.awssdk.services.s3.model.PutObjectResponse - Returns
metadata pertaining to the put object operation.
     */
    public PutObjectResponse putObjectFromStreamCrt(S3AsyncClient s33CrtAsyncClient,
String bucketName, String key) {

        // AsyncExampleUtils.randomString() returns a random string up to 100
characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
randomString.length());
        InputStream inputStream = new ByteArrayInputStream(randomString.getBytes());

        // Executor required to handle reading from the InputStream on a separate
thread so the main upload is not blocked.
        ExecutorService executor = Executors.newSingleThreadExecutor();
        // Specify `null` for the content length when you don't know the content
length.
        AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream, null,
executor);

        CompletableFuture<PutObjectResponse> responseFuture =
                s33CrtAsyncClient.putObject(r -> r.bucket(bucketName).key(key),
body);

        PutObjectResponse response = responseFuture.join(); // Wait for the
response.
```

```
        logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        executor.shutdown();
        return response;
    }
}
```

使用标准的启用了分段上传的异步 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.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

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

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket-" + UUID.randomUUID(); // Change
 bucket name.
        String key = UUID.randomUUID().toString();

        AsyncExampleUtils.createBucket(bucketName);
        try {
            PutObjectFromStreamAsyncMp example = new PutObjectFromStreamAsyncMp();
            S3AsyncClient s3AsyncClientMp =
 S3AsyncClient.builder().multipartEnabled(true).build();
            PutObjectResponse putObjectResponse =
 example.putObjectFromStreamMp(s3AsyncClientMp, bucketName, key);
            logger.info("Object {} etag: {}", key, putObjectResponse.eTag());
            logger.info("Object {} uploaded to bucket {}.", key, bucketName);
```

```
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
        } finally {
            AsyncExampleUtils.deleteObject(bucketName, key);
            AsyncExampleUtils.deleteBucket(bucketName);
        }
    }

    /**
     * @param s3AsyncClientMp - To upload content from a stream of unknown size, use
 can the S3 asynchronous client with multipart enabled.
     * @param bucketName - The name of the bucket.
     * @param key - The name of the object.
     * @return software.amazon.awssdk.services.s3.model.PutObjectResponse - Returns
 metadata pertaining to the put object operation.
     */
    public PutObjectResponse putObjectFromStreamMp(S3AsyncClient s3AsyncClientMp,
 String bucketName, String key) {

        // AsyncExampleUtils.randomString() returns a random string up to 100
 characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
 randomString.length());
        InputStream inputStream = new ByteArrayInputStream(randomString.getBytes());

        // Executor required to handle reading from the InputStream on a separate
 thread so the main upload is not blocked.
        ExecutorService executor = Executors.newSingleThreadExecutor();
        // Specify `null` for the content length when you don't know the content
 length.
        AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream, null,
 executor);

        CompletableFuture<PutObjectResponse> responseFuture =
                s3AsyncClientMp.putObject(r -> r.bucket(bucketName).key(key), body);

        PutObjectResponse response = responseFuture.join(); // Wait for the
 response.
        logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        executor.shutdown();
        return response;
    }
}
```

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.exception.SdkException;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedUpload;
import software.amazon.awssdk.transfer.s3.model.Upload;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.UUID;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

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

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket" + UUID.randomUUID();
        String key = UUID.randomUUID().toString();

        AsyncExampleUtils.createBucket(bucketName);
        try {
            UploadStream example = new UploadStream();
            CompletedUpload completedUpload =
 example.uploadStream(S3TransferManager.create(), bucketName, key);
            logger.info("Object {} etag: {}", key,
 completedUpload.response().eTag());
            logger.info("Object {} uploaded to bucket {}.", key, bucketName);
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
        } finally {
            AsyncExampleUtils.deleteObject(bucketName, key);
            AsyncExampleUtils.deleteBucket(bucketName);
        }
    }
```

```
    /**
     * @param transferManager - To upload content from a stream of unknown size, you
 can use the S3TransferManager based on the AWS CRT-based S3 client.
     * @param bucketName - The name of the bucket.
     * @param key - The name of the object.
     * @return - software.amazon.awssdk.transfer.s3.model.CompletedUpload - The
 result of the completed upload.
     */
    public CompletedUpload uploadStream(S3TransferManager transferManager, String
 bucketName, String key) {

        // AsyncExampleUtils.randomString() returns a random string up to 100
 characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
 randomString.length());
        InputStream inputStream = new ByteArrayInputStream(randomString.getBytes());

        // Executor required to handle reading from the InputStream on a separate
 thread so the main upload is not blocked.
        ExecutorService executor = Executors.newSingleThreadExecutor();
        // Specify `null` for the content length when you don't know the content
 length.
        AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream, null,
 executor);

        Upload upload = transferManager.upload(builder -> builder
                .requestBody(body)
                .putObjectRequest(req -> req.bucket(bucketName).key(key))
                .build());

        CompletedUpload completedUpload = upload.completionFuture().join();
        executor.shutdown();
        return completedUpload;
    }
}
```

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。

适用于 Java 的 SDK 2.x

> **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。

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)));
    }
```

在基于AWS CRT 的 S3 客户端上使用 S3 Transfer Manager，以在内容大小超过阈值时透明地执行分段上传。默认阈值大小为 8 MB。

您可以为要使用的开发工具包指定校验和算法。默认情况下，SDK 使用该 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《AWS SDK for Java 2.x API Reference》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。

适用于 Java 的 SDK 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);
    }
}
```

使用适用于 Java 的 SDK 2.x 的 Amazon S3 Control 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。

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 基本功能
- 操作

开始使用

Hello Amazon S3 Control

以下代码示例演示如何开始使用 Amazon S3 Control。

适用于 Java 的 SDK 2.x

> **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。

```
import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryMode;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlAsyncClient;
import software.amazon.awssdk.services.s3control.model.JobListDescriptor;
import software.amazon.awssdk.services.s3control.model.JobStatus;
import software.amazon.awssdk.services.s3control.model.ListJobsRequest;
import software.amazon.awssdk.services.s3control.paginators.ListJobsPublisher;
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this example:
 * <p/>
```

```
 * The SDK must be able to authenticate AWS requests on your behalf. If you have not
 configured
 * authentication for SDKs and tools,see https://docs.aws.amazon.com/sdkref/latest/
guide/access.html in the AWS SDKs and Tools Reference Guide.
 * <p/>
 * You must have a runtime environment configured with the Java SDK.
 * See https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html in
 the Developer Guide if this is not set up.
 */
public class HelloS3Batch {
    private static S3ControlAsyncClient asyncClient;

    public static void main(String[] args) {
        S3BatchActions actions = new S3BatchActions();
        String accountId = actions.getAccountId();
        try {
            listBatchJobsAsync(accountId)
                .exceptionally(ex -> {
                    System.err.println("List batch jobs failed: " +
 ex.getMessage());
                    return null;
                })
                .join();

        } catch (CompletionException ex) {
            System.err.println("Failed to list batch jobs: " + ex.getMessage());
        }
    }

    /**
     * Retrieves the asynchronous S3 Control client instance.
     * <p>
     * This method creates and returns a singleton instance of the {@link
 S3ControlAsyncClient}. If the instance
     * has not been created yet, it will be initialized with the following
 configuration:
     * <ul>
     *   <li>Maximum concurrency: 100</li>
     *   <li>Connection timeout: 60 seconds</li>
     *   <li>Read timeout: 60 seconds</li>
     *   <li>Write timeout: 60 seconds</li>
     *   <li>API call timeout: 2 minutes</li>
     *   <li>API call attempt timeout: 90 seconds</li>
     *   <li>Retry policy: 3 retries</li>
```

```
     *   <li>Region: US_EAST_1</li>
     *   <li>Credentials provider: {@link EnvironmentVariableCredentialsProvider}</
li>
     * </ul>
     *
     * @return the asynchronous S3 Control client instance
     */
    private static S3ControlAsyncClient getAsyncClient() {
        if (asyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            asyncClient = S3ControlAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return asyncClient;
    }

    /**
     * Asynchronously lists batch jobs that have completed for the specified
 account.
     *
     * @param accountId the ID of the account to list jobs for
     * @return a CompletableFuture that completes when the job listing operation is
 finished
     */
    public static CompletableFuture<Void> listBatchJobsAsync(String accountId) {
        ListJobsRequest jobsRequest = ListJobsRequest.builder()
            .jobStatuses(JobStatus.COMPLETE)
            .accountId(accountId)
```

```
            .maxResults(10)
            .build();

        ListJobsPublisher publisher =
getAsyncClient().listJobsPaginator(jobsRequest);
        return publisher.subscribe(response -> {
            List<JobListDescriptor> jobs = response.jobs();
            for (JobListDescriptor job : jobs) {
                System.out.println("The job id is " + job.jobId());
                System.out.println("The job priority is " + job.priority());
            }
        }).thenAccept(response -> {
            System.out.println("Listing batch jobs completed");
        }).exceptionally(ex -> {
            System.err.println("Failed to list batch jobs: " + ex.getMessage());
            throw new RuntimeException(ex);
        });
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ListJobs](#)中的。

基本功能

了解基本功能

以下代码示例演示如何了解 Amazon S3 Control 的核心操作。

适用于 Java 的 SDK 2.x

> **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。

了解核心操作。

```
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 = "amzn-s3-demo-bucket-" + UUID.randomUUID(); // Change
 bucket name.
        actions.createBucket(bucketName);
        String reportBucketName = "arn:aws:s3:::"+bucketName;
        String manifestLocation = "arn:aws:s3:::"+bucketName+"/job-manifest.csv";
        System.out.println("Populate the bucket with the required files.");
        String[] fileNames = {"job-manifest.csv", "object-key-1.txt", "object-
key-2.txt", "object-key-3.txt", "object-key-4.txt"};
        actions.uploadFilesToBucket(bucketName, fileNames, actions);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create a S3 Batch Job");
        System.out.println("This job tags all objects listed in the manifest file
 with tags");
        waitForInputToContinue(scanner);
        String jobId ;
        try {
            jobId = actions.createS3JobAsync(accountId, iamRoleArn,
 manifestLocation, reportBucketName, uuid).join();
            System.out.println("The Job id is " + jobId);

        } catch (S3Exception e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
```

```
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Update an existing S3 Batch Operations job's
priority");
        System.out.println("""
             In this step, we modify the job priority value. The higher the number,
the higher the priority.
             So, a job with a priority of `30` would have a higher priority than a
job with
             a priority of `20`. This is a common way to represent the priority of a
task
             or job, with higher numbers indicating a higher priority.

             Ensure that the job status allows for priority updates. Jobs in
certain
             states (e.g., Cancelled, Failed, or Completed) cannot have their
priorities
             updated. Only jobs in the Active or Suspended state typically allow
priority
             updates.
             """);

        try {
            actions.updateJobPriorityAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Update job priority failed: " +
ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to update job priority: " + ex.getMessage());
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Cancel the S3 Batch job");
        System.out.print("Do you want to cancel the Batch job? (y/n): ");
        String cancelAns = scanner.nextLine();
        if (cancelAns != null && cancelAns.trim().equalsIgnoreCase("y")) {
```

```
            try {
                actions.cancelJobAsync(jobId, accountId)
                    .exceptionally(ex -> {
                        System.err.println("Cancel job failed: " + ex.getMessage());
                        return null;
                    })
                    .join();
            } catch (CompletionException ex) {
                System.err.println("Failed to cancel job: " + ex.getMessage());
            }
        } else {
            System.out.println("Job " +jobId +" was not canceled.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Describe the job that was just created");
        waitForInputToContinue(scanner);
        try {
            actions.describeJobAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Describe job failed: " + ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to describe job: " + ex.getMessage());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Describe the tags associated with the job");
        waitForInputToContinue(scanner);
        try {
            actions.getJobTagsAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Get job tags failed: " + ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to get job tags: " + ex.getMessage());
        }
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("6. Update Batch Job Tags");
        waitForInputToContinue(scanner);
        try {
            actions.putJobTaggingAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Put job tagging failed: " +
ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to put job tagging: " + ex.getMessage());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the Amazon S3 Batch job tagging.");
        System.out.print("Do you want to delete Batch job tagging? (y/n)");
        String delAns = scanner.nextLine();
        if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
            try {
                actions.deleteBatchJobTagsAsync(jobId, accountId)
                    .exceptionally(ex -> {
                        System.err.println("Delete batch job tags failed: " +
ex.getMessage());
                        return null;
                    })
                    .join();
            } catch (CompletionException ex) {
                System.err.println("Failed to delete batch job tags: " +
ex.getMessage());
            }
        } else {
            System.out.println("Tagging was not deleted.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.print("Do you want to delete the AWS resources used in this
scenario? (y/n)");
        String delResAns = scanner.nextLine();
        if (delResAns != null && delResAns.trim().equalsIgnoreCase("y")) {
```

```
            actions.deleteFilesFromBucket(bucketName, fileNames, actions);
            actions.deleteBucketFolderAsync(bucketName);
            actions.deleteBucket(bucketName)
                .thenRun(() -> System.out.println("Bucket deletion completed"))
                .exceptionally(ex -> {
                    System.err.println("Error occurred: " + ex.getMessage());
                    return null;
                });
            CloudFormationHelper.destroyCloudFormationStack(STACK_NAME);
        } else {
            System.out.println("The AWS resources were not deleted.");
        }
        System.out.println("The Amazon S3 Batch scenario has successfully
 completed.");
        System.out.println(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println();
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println();
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }


}
```

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)
                .build();
        }
        return asyncClient;
    }

    private static S3AsyncClient getS3AsyncClient() {
        if (asyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            s3AsyncClient = S3AsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return s3AsyncClient;
    }


    /**
     * Cancels a job asynchronously.
     *
     * @param jobId The ID of the job to be canceled.
     * @param accountId The ID of the account associated with the job.
     * @return A {@link CompletableFuture} that completes when the job status has
been updated to "CANCELLED".
     *         If an error occurs during the update, the returned future will
complete exceptionally.
     */
    public CompletableFuture<Void> cancelJobAsync(String jobId, String accountId) {
```

```
        UpdateJobStatusRequest updateJobStatusRequest =
UpdateJobStatusRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .requestedJobStatus(String.valueOf(JobStatus.CANCELLED))
            .build();

        return asyncClient.updateJobStatus(updateJobStatusRequest)
            .thenAccept(updateJobStatusResponse -> {
                System.out.println("Job status updated to: " +
updateJobStatusResponse.status());
            })
            .exceptionally(ex -> {
                System.err.println("Failed to cancel job: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }

    /**
     * Updates the priority of a job asynchronously.
     *
     * @param jobId     the ID of the job to update
     * @param accountId the ID of the account associated with the job
     * @return a {@link CompletableFuture} that represents the asynchronous
operation, which completes when the job priority has been updated or an error has
occurred
     */
    public CompletableFuture<Void> updateJobPriorityAsync(String jobId, String
accountId) {
        UpdateJobPriorityRequest priorityRequest =
UpdateJobPriorityRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .priority(60)
            .build();

        CompletableFuture<Void> future = new CompletableFuture<>();
        getAsyncClient().updateJobPriority(priorityRequest)
            .thenAccept(response -> {
                System.out.println("The job priority was updated");
                future.complete(null); // Complete the CompletableFuture on
successful execution
            })
            .exceptionally(ex -> {
```

```
                System.err.println("Failed to update job priority: " +
ex.getMessage());
                future.completeExceptionally(ex); // Complete the CompletableFuture
exceptionally on error
                return null; // Return null to handle the exception
            });

        return future;
    }

    /**
     * Asynchronously retrieves the tags associated with a specific job in an AWS
account.
     *
     * @param jobId     the ID of the job for which to retrieve the tags
     * @param accountId the ID of the AWS account associated with the job
     * @return a {@link CompletableFuture} that completes when the job tags have
been retrieved, or with an exception if the operation fails
     * @throws RuntimeException if an error occurs while retrieving the job tags
     */
    public CompletableFuture<Void> getJobTagsAsync(String jobId, String accountId) {
        GetJobTaggingRequest request = GetJobTaggingRequest.builder()
            .jobId(jobId)
            .accountId(accountId)
            .build();

        return asyncClient.getJobTagging(request)
            .thenAccept(response -> {
                List<S3Tag> tags = response.tags();
                if (tags.isEmpty()) {
                    System.out.println("No tags found for job ID: " + jobId);
                } else {
                    for (S3Tag tag : tags) {
                        System.out.println("Tag key is: " + tag.key());
                        System.out.println("Tag value is: " + tag.value());
                    }
                }
            })
            .exceptionally(ex -> {
                System.err.println("Failed to get job tags: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }
```

```
    /**
     * Asynchronously deletes the tags associated with a specific batch job.
     *
     * @param jobId     The ID of the batch job whose tags should be deleted.
     * @param accountId The ID of the account associated with the batch job.
     * @return A CompletableFuture that completes when the job tags have been
successfully deleted, or an exception is thrown if the deletion fails.
     */
    public CompletableFuture<Void> deleteBatchJobTagsAsync(String jobId, String
accountId) {
        DeleteJobTaggingRequest jobTaggingRequest =
DeleteJobTaggingRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .build();

        return asyncClient.deleteJobTagging(jobTaggingRequest)
            .thenAccept(response -> {
                System.out.println("You have successfully deleted " + jobId + "
tagging.");
            })
            .exceptionally(ex -> {
                System.err.println("Failed to delete job tags: " + ex.getMessage());
                throw new RuntimeException(ex);
            });
    }

    /**
     * Asynchronously describes the specified job.
     *
     * @param jobId     the ID of the job to describe
     * @param accountId the ID of the AWS account associated with the job
     * @return a {@link CompletableFuture} that completes when the job description
is available
     * @throws RuntimeException if an error occurs while describing the job
     */
    public CompletableFuture<Void> describeJobAsync(String jobId, String accountId)
{
        DescribeJobRequest jobRequest = DescribeJobRequest.builder()
            .jobId(jobId)
            .accountId(accountId)
            .build();

        return getAsyncClient().describeJob(jobRequest)
```

```
            .thenAccept(response -> {
                System.out.println("Job ID: " + response.job().jobId());
                System.out.println("Description: " + response.job().description());
                System.out.println("Status: " + response.job().statusAsString());
                System.out.println("Role ARN: " + response.job().roleArn());
                System.out.println("Priority: " + response.job().priority());
                System.out.println("Progress Summary: " +
response.job().progressSummary());

                // Print out details about the job manifest.
                JobManifest manifest = response.job().manifest();
                System.out.println("Manifest Location: " +
manifest.location().objectArn());
                System.out.println("Manifest ETag: " + manifest.location().eTag());

                // Print out details about the job operation.
                JobOperation operation = response.job().operation();
                if (operation.s3PutObjectTagging() != null) {
                    System.out.println("Operation: S3 Put Object Tagging");
                    System.out.println("Tag Set: " +
operation.s3PutObjectTagging().tagSet());
                }

                // Print out details about the job report.
                JobReport report = response.job().report();
                System.out.println("Report Bucket: " + report.bucket());
                System.out.println("Report Prefix: " + report.prefix());
                System.out.println("Report Format: " + report.format());
                System.out.println("Report Enabled: " + report.enabled());
                System.out.println("Report Scope: " + report.reportScopeAsString());
            })
            .exceptionally(ex -> {
                System.err.println("Failed to describe job: " + ex.getMessage());
                throw new RuntimeException(ex);
            });
    }

    /**
     * Creates an asynchronous S3 job using the AWS Java SDK.
     *
     * @param accountId         the AWS account ID associated with the job
     * @param iamRoleArn        the ARN of the IAM role to be used for the job
     * @param manifestLocation  the location of the job manifest file in S3
     * @param reportBucketName  the name of the S3 bucket to store the job report
```

```
     * @param uuid              a unique identifier for the job
     * @return a CompletableFuture that represents the asynchronous creation of the
 S3 job.
     *         The CompletableFuture will return the job ID if the job is created
 successfully,
     *         or throw an exception if there is an error.
     */
    public CompletableFuture<String> createS3JobAsync(String accountId, String
 iamRoleArn,
                                                       String manifestLocation,
 String reportBucketName, String uuid) {

        String[] bucketName = new String[]{""};
        String[] parts = reportBucketName.split(":::");
        if (parts.length > 1) {
            bucketName[0] = parts[1];
        } else {
            System.out.println("The input string does not contain the expected
 format.");
        }

        return CompletableFuture.supplyAsync(() -> getETag(bucketName[0], "job-
manifest.csv"))
            .thenCompose(eTag -> {
                  ArrayList<S3Tag> tagSet = new ArrayList<>();
                S3Tag s3Tag = S3Tag.builder()
                    .key("keyOne")
                    .value("ValueOne")
                    .build();
                S3Tag s3Tag2 = S3Tag.builder()
                    .key("keyTwo")
                    .value("ValueTwo")
                    .build();
                tagSet.add(s3Tag);
                tagSet.add(s3Tag2);

                S3SetObjectTaggingOperation objectTaggingOperation =
 S3SetObjectTaggingOperation.builder()
                    .tagSet(tagSet)
                    .build();

                JobOperation jobOperation = JobOperation.builder()
                    .s3PutObjectTagging(objectTaggingOperation)
                    .build();
```

```
                JobManifestLocation jobManifestLocation =
JobManifestLocation.builder()
                    .objectArn(manifestLocation)
                    .eTag(eTag)
                    .build();

                JobManifestSpec manifestSpec = JobManifestSpec.builder()
                    .fieldsWithStrings("Bucket", "Key")
                    .format("S3BatchOperations_CSV_20180820")
                    .build();

                JobManifest jobManifest = JobManifest.builder()
                    .spec(manifestSpec)
                    .location(jobManifestLocation)
                    .build();

                JobReport jobReport = JobReport.builder()
                    .bucket(reportBucketName)
                    .prefix("reports")
                    .format("Report_CSV_20180820")
                    .enabled(true)
                    .reportScope("AllTasks")
                    .build();

                CreateJobRequest jobRequest = CreateJobRequest.builder()
                    .accountId(accountId)
                    .description("Job created using the AWS Java SDK")
                    .manifest(jobManifest)
                    .operation(jobOperation)
                    .report(jobReport)
                    .priority(42)
                    .roleArn(iamRoleArn)
                    .clientRequestToken(uuid)
                    .confirmationRequired(false)
                    .build();

                // Create the job asynchronously.
                 return getAsyncClient().createJob(jobRequest)
                    .thenApply(CreateJobResponse::jobId);
                 })
                 .handle((jobId, ex) -> {
                    if (ex != null) {
```

```
                    Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                    if (cause instanceof S3ControlException) {
                        throw new CompletionException(cause);
                    } else {
                        throw new RuntimeException(cause);
                    }
                }
                return jobId;
            });
    }

    /**
     * Retrieves the ETag (Entity Tag) for an object stored in an Amazon S3 bucket.
     *
     * @param bucketName the name of the Amazon S3 bucket where the object is stored
     * @param key the key (file name) of the object in the Amazon S3 bucket
     * @return the ETag of the object
     */
    public String getETag(String bucketName, String key) {
        S3Client s3Client = S3Client.builder()
            .region(Region.US_EAST_1)
            .build();

        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        HeadObjectResponse headObjectResponse =
s3Client.headObject(headObjectRequest);
        return headObjectResponse.eTag();
    }

    /**
     * Asynchronously adds tags to a job in the system.
     *
     * @param jobId     the ID of the job to add tags to
     * @param accountId the account ID associated with the job
     * @return a CompletableFuture that completes when the tagging operation is
finished
     */
    public CompletableFuture<Void> putJobTaggingAsync(String jobId, String
accountId) {
```

```
        S3Tag departmentTag = S3Tag.builder()
            .key("department")
            .value("Marketing")
            .build();

        S3Tag fiscalYearTag = S3Tag.builder()
            .key("FiscalYear")
            .value("2020")
            .build();

        PutJobTaggingRequest putJobTaggingRequest = PutJobTaggingRequest.builder()
            .jobId(jobId)
            .accountId(accountId)
            .tags(departmentTag, fiscalYearTag)
            .build();

        return asyncClient.putJobTagging(putJobTaggingRequest)
            .thenRun(() -> {
                System.out.println("Additional Tags were added to job " + jobId);
            })
            .exceptionally(ex -> {
                System.err.println("Failed to add tags to job: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }

    // Setup the S3 bucket required for this scenario.
    /**
     * Creates an Amazon S3 bucket with the specified name.
     *
     * @param bucketName the name of the S3 bucket to create
     * @throws S3Exception if there is an error creating the bucket
     */
    public void createBucket(String bucketName) {
        try {
            S3Client s3Client = S3Client.builder()
                .region(Region.US_EAST_1)
                .build();


            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                .bucket(bucketName)
                .build();
```

```
            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Uploads a file to an Amazon S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to upload the file to
     * @param fileName the name of the file to be uploaded
     * @throws RuntimeException if an error occurs during the file upload
     */
    public void populateBucket(String bucketName, String fileName) {
        // Define the path to the directory.
        Path filePath = Paths.get("src/main/resources/batch/",
fileName).toAbsolutePath();
        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(fileName)
            .build();

        CompletableFuture<PutObjectResponse> future =
getS3AsyncClient().putObject(putOb, AsyncRequestBody.fromFile(filePath));
        future.whenComplete((result, ex) -> {
            if (ex != null) {
                System.err.println("Error uploading file: " + ex.getMessage());
            } else {
                System.out.println("Successfully placed " + fileName + " into bucket
" + bucketName);
            }
        }).join();
```

```
    }


    // Update the bucketName in CSV.
    public void updateCSV(String newValue) {
        Path csvFilePath = Paths.get("src/main/resources/batch/job-
manifest.csv").toAbsolutePath();
        try {
            // Read all lines from the CSV file.
            List<String> lines = Files.readAllLines(csvFilePath);

            // Update the first value in each line.
            List<String> updatedLines = lines.stream()
                .map(line -> {
                    String[] parts = line.split(",");
                    parts[0] = newValue;
                    return String.join(",", parts);
                })
                .collect(Collectors.toList());

            // Write the updated lines back to the CSV file
            Files.write(csvFilePath, updatedLines);
            System.out.println("CSV file updated successfully.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    /**
     * Deletes an object from an Amazon S3 bucket asynchronously.
     *
     * @param bucketName The name of the S3 bucket where the object is stored.
     * @param objectName The name of the object to be deleted.
     * @return A {@link CompletableFuture} that completes when the object has been
 deleted,
     *         or throws a {@link RuntimeException} if an error occurs during the
 deletion.
     */
    public CompletableFuture<Void> deleteBucketObjects(String bucketName, String
 objectName) {
        ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();
        toDelete.add(ObjectIdentifier.builder()
            .key(objectName)
            .build());
```

```
        DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
            .bucket(bucketName)
            .delete(Delete.builder()
                .objects(toDelete).build())
            .build();

        return getS3AsyncClient().deleteObjects(dor)
            .thenAccept(result -> {
                System.out.println("The object was deleted!");
            })
            .exceptionally(ex -> {
                throw new RuntimeException("Error deleting object: " +
ex.getMessage(), ex);
            });
    }

    /**
     * Deletes a folder and all its contents asynchronously from an Amazon S3
bucket.
     *
     * @param bucketName the name of the S3 bucket containing the folder to be
deleted
     * @return a {@link CompletableFuture} that completes when the folder and its
contents have been deleted
     * @throws RuntimeException if any error occurs during the deletion process
     */
    public void deleteBucketFolderAsync(String bucketName) {
        String folderName = "reports/";
        ListObjectsV2Request request = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .prefix(folderName)
            .build();

        CompletableFuture<ListObjectsV2Response> listObjectsFuture =
getS3AsyncClient().listObjectsV2(request);
        listObjectsFuture.thenCompose(response -> {
            List<CompletableFuture<DeleteObjectResponse>> deleteFutures =
response.contents().stream()
                .map(obj -> {
                    DeleteObjectRequest deleteRequest =
DeleteObjectRequest.builder()
                        .bucket(bucketName)
                        .key(obj.key())
```

```
                        .build();
                    return getS3AsyncClient().deleteObject(deleteRequest)
                        .thenApply(deleteResponse -> {
                            System.out.println("Deleted object: " + obj.key());
                            return deleteResponse;
                        });
                })
                .collect(Collectors.toList());

            return CompletableFuture.allOf(deleteFutures.toArray(new
CompletableFuture[0]))
                .thenCompose(v -> {
                    // Delete the folder.
                    DeleteObjectRequest deleteRequest =
DeleteObjectRequest.builder()
                        .bucket(bucketName)
                        .key(folderName)
                        .build();
                    return getS3AsyncClient().deleteObject(deleteRequest)
                        .thenApply(deleteResponse -> {
                            System.out.println("Deleted folder: " + folderName);
                            return deleteResponse;
                        });
                });
        }).join();
    }

    /**
     * Deletes an Amazon S3 bucket.
     *
     * @param bucketName the name of the bucket to delete
     * @return a {@link CompletableFuture} that completes when the bucket has been
deleted, or exceptionally if there is an error
     * @throws RuntimeException if there is an error deleting the bucket
     */
    public CompletableFuture<Void> deleteBucket(String bucketName) {
        S3AsyncClient s3Client = getS3AsyncClient();
        return s3Client.deleteBucket(DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build())
            .thenAccept(deleteBucketResponse -> {
                System.out.println(bucketName + " was deleted");
            })
            .exceptionally(ex -> {
```

```
                // Handle the exception or rethrow it.
                throw new RuntimeException("Failed to delete bucket: " + bucketName,
ex);
            });
    }

    /**
     * Uploads a set of files to an Amazon S3 bucket.
     *
     * @param bucketName the name of the S3 bucket to upload the files to
     * @param fileNames an array of file names to be uploaded
     * @param actions an instance of {@link S3BatchActions} that provides the
implementation for the necessary S3 operations
     * @throws IOException if there's an error creating the text files or uploading
the files to the S3 bucket
     */
    public static void uploadFilesToBucket(String bucketName, String[] fileNames,
S3BatchActions actions) throws IOException {
        actions.updateCSV(bucketName);
        createTextFiles(fileNames);
        for (String fileName : fileNames) {
            actions.populateBucket(bucketName, fileName);
        }
        System.out.println("All files are placed in the S3 bucket " + bucketName);
    }

    /**
     * Deletes the specified files from the given S3 bucket.
     *
     * @param bucketName the name of the S3 bucket
     * @param fileNames an array of file names to be deleted from the bucket
     * @param actions the S3BatchActions instance to be used for the file deletion
     * @throws IOException if an I/O error occurs during the file deletion
     */
    public void deleteFilesFromBucket(String bucketName, String[] fileNames,
S3BatchActions actions) throws IOException {
        for (String fileName : fileNames) {
                    actions.deleteBucketObjects(bucketName, fileName)
                   .thenRun(() -> System.out.println("Object deletion completed"))
                   .exceptionally(ex -> {
                       System.err.println("Error occurred: " + ex.getMessage());
                       return null;
                   });
        }
```

```
        System.out.println("All files have been deleted from the bucket " +
bucketName);
    }

    public static void createTextFiles(String[] fileNames) {
        String currentDirectory = System.getProperty("user.dir");
        String directoryPath = currentDirectory + "\\src\\main\\resources\\batch";
        Path path = Paths.get(directoryPath);

        try {
            // Create the directory if it doesn't exist.
            if (Files.notExists(path)) {
                Files.createDirectories(path);
                System.out.println("Created directory: " + path.toString());
            } else {
                System.out.println("Directory already exists: " + path.toString());
            }

            for (String fileName : fileNames) {
                // Check if the file is a .txt file.
                if (fileName.endsWith(".txt")) {
                    // Define the path for the new file.
                    Path filePath = path.resolve(fileName);
                    System.out.println("Attempting to create file: " +
filePath.toString());

                    // Create and write content to the new file.
                    Files.write(filePath, "This is a test".getBytes());

                    // Verify the file was created.
                    if (Files.exists(filePath)) {
                        System.out.println("Successfully created file: " +
filePath.toString());
                    } else {
                        System.out.println("Failed to create file: " +
filePath.toString());
                    }
                }
            }

        } catch (IOException e) {
            System.err.println("An error occurred: " + e.getMessage());
            e.printStackTrace();
        }
```

```
    }

    public String getAccountId() {
        StsClient stsClient = StsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        GetCallerIdentityResponse callerIdentityResponse =
 stsClient.getCallerIdentity();
        return callerIdentityResponse.account();
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - CreateJob
  - DeleteJobTagging
  - DescribeJob
  - GetJobTagging
  - ListJobs
  - PutJobTagging
  - UpdateJobPriority
  - UpdateJobStatus

操作

`CreateJob`

以下代码示例演示了如何使用 `CreateJob`。

适用于 Java 的 SDK 2.x

> **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。

创建异步 S3 作业。

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

创建合规性保留作业。

```
    /**
     * Creates a compliance retention job in Amazon S3 Control.
     * <p>
     * A compliance retention job in Amazon S3 Control is a feature that allows you
 to
     * set a retention period for objects stored in an S3 bucket.
     * This feature is particularly useful for organizations that need to comply
 with
     * regulatory requirements or internal policies that mandate the retention of
 data for
     * a specific duration.
     *
     * @param s3ControlClient The S3ControlClient instance to use for the API call.
     * @return The job ID of the created compliance retention job.
     */
    public static String createComplianceRetentionJob(final S3ControlClient
 s3ControlClient, String roleArn, String bucketName, String accountId) {
        final String manifestObjectArn = "arn:aws:s3:::amzn-s3-demo-manifest-bucket/
compliance-objects-manifest.csv";
```

```
        final String manifestObjectVersionId = "your-object-version-Id";

        Instant jan2025 = Instant.parse("2025-01-01T00:00:00Z");
        JobOperation jobOperation = JobOperation.builder()
            .s3PutObjectRetention(S3SetObjectRetentionOperation.builder()
                .retention(S3Retention.builder()
                    .mode(S3ObjectLockRetentionMode.COMPLIANCE)
                    .retainUntilDate(jan2025)
                    .build())
                .build())
            .build();

        JobManifestLocation manifestLocation = JobManifestLocation.builder()
            .objectArn(manifestObjectArn)
            .eTag(manifestObjectVersionId)
            .build();

        JobManifestSpec manifestSpec = JobManifestSpec.builder()
            .fieldsWithStrings("Bucket", "Key")
            .format("S3BatchOperations_CSV_20180820")
            .build();

        JobManifest manifestToPublicApi = JobManifest.builder()
            .location(manifestLocation)
            .spec(manifestSpec)
            .build();

        // Report details.
        final String jobReportBucketArn = "arn:aws:s3:::" + bucketName;
        final String jobReportPrefix = "reports/compliance-objects-bops";

        JobReport jobReport = JobReport.builder()
            .enabled(true)
            .reportScope(JobReportScope.ALL_TASKS)
            .bucket(jobReportBucketArn)
            .prefix(jobReportPrefix)
            .format(JobReportFormat.REPORT_CSV_20180820)
            .build();

        final Boolean requiresConfirmation = true;
        final int priority = 10;
        CreateJobRequest request = CreateJobRequest.builder()
            .accountId(accountId)
            .description("Set compliance retain-until to 1 Jan 2025")
```

```
            .manifest(manifestToPublicApi)
            .operation(jobOperation)
            .priority(priority)
            .roleArn(roleArn)
            .report(jobReport)
            .confirmationRequired(requiresConfirmation)
            .build();

        // Create the job and get the result.
        CreateJobResponse result = s3ControlClient.createJob(request);
        return result.jobId();
    }
```

创建法定保留作业。

```
    /**
     * Creates a compliance retention job in Amazon S3 Control.
     * <p>
     * A compliance retention job in Amazon S3 Control is a feature that allows you
 to
     * set a retention period for objects stored in an S3 bucket.
     * This feature is particularly useful for organizations that need to comply
 with
     * regulatory requirements or internal policies that mandate the retention of
 data for
     * a specific duration.
     *
     * @param s3ControlClient The S3ControlClient instance to use for the API call.
     * @return The job ID of the created compliance retention job.
     */
    public static String createComplianceRetentionJob(final S3ControlClient
 s3ControlClient, String roleArn, String bucketName, String accountId) {
        final String manifestObjectArn = "arn:aws:s3:::amzn-s3-demo-manifest-bucket/
compliance-objects-manifest.csv";
        final String manifestObjectVersionId = "your-object-version-Id";

        Instant jan2025 = Instant.parse("2025-01-01T00:00:00Z");
        JobOperation jobOperation = JobOperation.builder()
            .s3PutObjectRetention(S3SetObjectRetentionOperation.builder()
                .retention(S3Retention.builder()
                    .mode(S3ObjectLockRetentionMode.COMPLIANCE)
```

```
                .retainUntilDate(jan2025)
                .build())
            .build())
        .build();

JobManifestLocation manifestLocation = JobManifestLocation.builder()
    .objectArn(manifestObjectArn)
    .eTag(manifestObjectVersionId)
    .build();

JobManifestSpec manifestSpec = JobManifestSpec.builder()
    .fieldsWithStrings("Bucket", "Key")
    .format("S3BatchOperations_CSV_20180820")
    .build();

JobManifest manifestToPublicApi = JobManifest.builder()
    .location(manifestLocation)
    .spec(manifestSpec)
    .build();

// Report details.
final String jobReportBucketArn = "arn:aws:s3:::" + bucketName;
final String jobReportPrefix = "reports/compliance-objects-bops";

JobReport jobReport = JobReport.builder()
    .enabled(true)
    .reportScope(JobReportScope.ALL_TASKS)
    .bucket(jobReportBucketArn)
    .prefix(jobReportPrefix)
    .format(JobReportFormat.REPORT_CSV_20180820)
    .build();

final Boolean requiresConfirmation = true;
final int priority = 10;
CreateJobRequest request = CreateJobRequest.builder()
    .accountId(accountId)
    .description("Set compliance retain-until to 1 Jan 2025")
    .manifest(manifestToPublicApi)
    .operation(jobOperation)
    .priority(priority)
    .roleArn(roleArn)
    .report(jobReport)
    .confirmationRequired(requiresConfirmation)
    .build();
```

```
        // Create the job and get the result.
        CreateJobResponse result = s3ControlClient.createJob(request);
        return result.jobId();
    }
```

创建新的治理保留作业。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateGovernanceRetentionJob {

    public static void main(String[]args) throws ParseException {
        final String usage = """

            Usage:
                <manifestObjectArn> <jobReportBucketArn> <roleArn> <accountId>
 <manifestObjectVersionId>

            Where:
                manifestObjectArn - The Amazon Resource Name (ARN) of the S3 object
 that contains the manifest file for the governance objects.\s
                bucketName - The ARN of the S3 bucket where the job report will be
 stored.
                roleArn - The ARN of the IAM role that will be used to perform the
 governance retention operation.
                accountId - Your AWS account Id.
                manifestObjectVersionId =  A unique value that is used as the `eTag`
 property of the `JobManifestLocation` object.
            """;

        if (args.length != 4) {
            System.out.println(usage);
            return;
        }
```

```
        String manifestObjectArn = args[0];
        String jobReportBucketArn = args[1];
        String roleArn = args[2];
        String accountId = args[3];
        String manifestObjectVersionId = args[4];

        S3ControlClient s3ControlClient = S3ControlClient.create();
        createGovernanceRetentionJob(s3ControlClient, manifestObjectArn,
jobReportBucketArn, roleArn, accountId, manifestObjectVersionId);
    }

    public static String createGovernanceRetentionJob(final S3ControlClient
s3ControlClient, String manifestObjectArn, String jobReportBucketArn, String
roleArn, String accountId, String manifestObjectVersionId) throws ParseException {
        final JobManifestLocation manifestLocation = JobManifestLocation.builder()
            .objectArn(manifestObjectArn)
            .eTag(manifestObjectVersionId)
            .build();

        final JobManifestSpec manifestSpec = JobManifestSpec.builder()
            .format(JobManifestFormat.S3_BATCH_OPERATIONS_CSV_20180820)
            .fields(Arrays.asList(JobManifestFieldName.BUCKET,
JobManifestFieldName.KEY))
            .build();

        final JobManifest manifestToPublicApi = JobManifest.builder()
            .location(manifestLocation)
            .spec(manifestSpec)
            .build();

        final String jobReportPrefix = "reports/governance-objects";
        final JobReport jobReport = JobReport.builder()
            .enabled(true)
            .reportScope(JobReportScope.ALL_TASKS)
            .bucket(jobReportBucketArn)
            .prefix(jobReportPrefix)
            .format(JobReportFormat.REPORT_CSV_20180820)
            .build();

        final SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
        final Date jan30th = format.parse("30/01/2025");

        final S3SetObjectRetentionOperation s3SetObjectRetentionOperation =
S3SetObjectRetentionOperation.builder()
```

```
            .retention(S3Retention.builder()
                .mode(S3ObjectLockRetentionMode.GOVERNANCE)
                .retainUntilDate(jan30th.toInstant())
                .build())
            .build();

        final JobOperation jobOperation = JobOperation.builder()
            .s3PutObjectRetention(s3SetObjectRetentionOperation)
            .build();

        final Boolean requiresConfirmation = true;
        final int priority = 10;

        final CreateJobRequest request = CreateJobRequest.builder()
            .accountId(accountId)
            .description("Put governance retention")
            .manifest(manifestToPublicApi)
            .operation(jobOperation)
            .priority(priority)
            .roleArn(roleArn)
            .report(jobReport)
            .confirmationRequired(requiresConfirmation)
            .build();

        final CreateJobResponse result = s3ControlClient.createJob(request);
        return result.jobId();
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateJob中的。

`DeleteJobTagging`

以下代码示例演示了如何使用 `DeleteJobTagging`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[DeleteJobTagging](#)中的。

`DescribeJob`

以下代码示例演示了如何使用 `DescribeJob`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeJob中的。

`GetJobTagging`

以下代码示例演示了如何使用 `GetJobTagging`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[GetJobTagging](#)中的。

`PutJobTagging`

以下代码示例演示了如何使用 `PutJobTagging`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutJobTagging中的。

`UpdateJobPriority`

以下代码示例演示了如何使用 `UpdateJobPriority`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[UpdateJobPriority](#)中的。

UpdateJobStatus

以下代码示例演示了如何使用 UpdateJobStatus。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考UpdateJobStatus中的。

使用适用于 Java 的 SDK 2.x 的 S3 目录存储桶示例

以下代码示例向您展示了如何使用 AWS SDK for Java 2.x 与 S3 Directory Buckets 配合使用来执行操作和实现常见场景。

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 基本功能
- 操作
- 场景

开始使用

Hello Amazon S3 目录存储桶

以下代码示例演示如何开始使用 Amazon S3 目录存储桶。

适用于 Java 的 SDK 2.x

> **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。

```
package com.example.s3.directorybucket;
```

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.BucketInfo;
import software.amazon.awssdk.services.s3.model.BucketType;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateBucketResponse;
import software.amazon.awssdk.services.s3.model.DataRedundancy;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsResponse;
import software.amazon.awssdk.services.s3.model.LocationInfo;
import software.amazon.awssdk.services.s3.model.LocationType;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.List;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;

/**
 * Before running this example:
 * <p>
 * The SDK must be able to authenticate AWS requests on your behalf. If you have
 * not configured
 * authentication for SDKs and tools, see
 * https://docs.aws.amazon.com/sdkref/latest/guide/access.html in the AWS SDKs
 * and Tools Reference Guide.
 * <p>
 * You must have a runtime environment configured with the Java SDK.
 * See
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html in
 * the Developer Guide if this is not set up.
 * <p>
 * To use S3 directory buckets, configure a gateway VPC endpoint. This is the
 * recommended method to enable directory bucket traffic without
 * requiring an internet gateway or NAT device. For more information on
 * configuring VPC gateway endpoints, visit
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-express-
networking.html#s3-express-networking-vpc-gateway.
```

```
 * <p>
 * Directory buckets are available in specific AWS Regions and Zones. For
 * details on Regions and Zones supporting directory buckets, see
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-express-
networking.html#s3-express-endpoints.
 */

public class HelloS3DirectoryBuckets {
    private static final Logger logger =
 LoggerFactory.getLogger(HelloS3DirectoryBuckets.class);

    public static void main(String[] args) {
        String bucketName = "test-bucket-" + System.currentTimeMillis() + "--usw2-
az1--x-s3";
        Region region = Region.US_WEST_2;
        String zone = "usw2-az1";
        S3Client s3Client = createS3Client(region);

        try {
            // Create the directory bucket
            createDirectoryBucket(s3Client, bucketName, zone);
            logger.info("Created bucket: {}", bucketName);

            // List all directory buckets
            List<String> bucketNames = listDirectoryBuckets(s3Client);
            bucketNames.forEach(name -> logger.info("Bucket Name: {}", name));
        } catch (S3Exception e) {
            logger.error("An error occurred during S3 operations: {} - Error code:
 {}",
                    e.awsErrorDetails().errorMessage(),
 e.awsErrorDetails().errorCode(), e);
        } finally {
            try {
                // Delete the created bucket
                deleteDirectoryBucket(s3Client, bucketName);
                logger.info("Deleted bucket: {}", bucketName);
            } catch (S3Exception e) {
                logger.error("Failed to delete the bucket due to S3 error: {} -
 Error code: {}",
                        e.awsErrorDetails().errorMessage(),
 e.awsErrorDetails().errorCode(), e);
            } catch (RuntimeException e) {
                logger.error("Failed to delete the bucket due to unexpected error:
 {}", e.getMessage(), e);
```

```
            } finally {
                s3Client.close();
            }
        }
    }

    /**
     * Creates a new S3 directory bucket in a specified Zone (For example, a
     * specified Availability Zone in this code example).
     *
     * @param s3Client   The S3 client used to create the bucket
     * @param bucketName The name of the bucket to be created
     * @param zone       The region where the bucket will be created
     * @throws S3Exception if there's an error creating the bucket
     */
    public static void createDirectoryBucket(S3Client s3Client, String bucketName,
String zone) throws S3Exception {
        logger.info("Creating bucket: {}", bucketName);

        CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
                .location(LocationInfo.builder()
                        .type(LocationType.AVAILABILITY_ZONE)
                        .name(zone).build())
                .bucket(BucketInfo.builder()
                        .type(BucketType.DIRECTORY)
                        .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                        .build())
                .build();
        try {
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .createBucketConfiguration(bucketConfiguration).build();
            CreateBucketResponse response = s3Client.createBucket(bucketRequest);
            logger.info("Bucket created successfully with location: {}",
response.location());
        } catch (S3Exception e) {
            logger.error("Error creating bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

```
    /**
     * Lists all S3 directory buckets.
     *
     * @param s3Client The S3 client used to interact with S3
     * @return A list of bucket names
     */
    public static List<String> listDirectoryBuckets(S3Client s3Client) {
        logger.info("Listing all directory buckets");

        try {
            // Create a ListBucketsRequest
            ListDirectoryBucketsRequest listBucketsRequest =
ListDirectoryBucketsRequest.builder().build();

            // Retrieve the list of buckets
            ListDirectoryBucketsResponse response =
s3Client.listDirectoryBuckets(listBucketsRequest);

            // Extract bucket names
            List<String> bucketNames = response.buckets().stream()
                    .map(Bucket::name)
                    .collect(Collectors.toList());

            return bucketNames;
        } catch (S3Exception e) {
            logger.error("Failed to list buckets: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }

    /**
     * Deletes the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the bucket to delete
     */
    public static void deleteDirectoryBucket(S3Client s3Client, String bucketName) {
        try {
            DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                    .bucket(bucketName)
                    .build();
            s3Client.deleteBucket(deleteBucketRequest);
```

```
        } catch (S3Exception e) {
            logger.error("Failed to delete bucket: " + bucketName + " - Error code:
 " + e.awsErrorDetails().errorCode(),
                    e);
            throw e;
        }
    }

}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - CreateBucket
  - ListDirectoryBuckets

基本功能

了解基本功能

以下代码示例展示了如何：

- 设置 VPC 和 VPC 端点。
- 设置策略、角色和用户，以使用 S3 目录存储桶和 S3 Express One Zone 存储类别。
- 创建两个 S3 客户端。
- 创建两个存储桶。
- 创建一个对象并复制它。
- 演示性能差异。
- 填充存储桶以显示按字典顺序的差异。
- 提示用户查看他们是否要清理资源。

适用于 Java 的 SDK 2.x

> **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。

运行一个交互式场景，演示 Amazon S3 的功能。

```
public class S3DirectoriesScenario {

    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    private static final Logger logger =
 LoggerFactory.getLogger(S3DirectoriesScenario.class);
    static Scanner scanner = new Scanner(System.in);

    private static S3AsyncClient mS3RegularClient;
    private static S3AsyncClient mS3ExpressClient;

    private static String mdirectoryBucketName;
    private static String mregularBucketName;

    private static String stackName = "cfn-stack-s3-express-basics--" +
 UUID.randomUUID();

    private static String regularUser = "";
    private static String vpcId = "";
    private static String expressUser = "";

    private static String vpcEndpointId = "";

    private static final S3DirectoriesActions s3DirectoriesActions = new
 S3DirectoriesActions();

    public static void main(String[] args) {
        try {
            s3ExpressScenario();
        } catch (RuntimeException e) {
            logger.info(e.getMessage());
        }
    }

    // Runs the scenario.
    private static void s3ExpressScenario() {
        logger.info(DASHES);
        logger.info("Welcome to the Amazon S3 Express Basics demo using AWS SDK for
 Java V2.");
        logger.info("""
            Let's get started! First, please note that S3 Express One Zone works
 best when working within the AWS infrastructure,
```

```
            specifically when working in the same Availability Zone (AZ). To see the
best results in this example and when you implement
            directory buckets into your infrastructure, it is best to put your
compute resources in the same AZ as your directory
            bucket.
            """);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        // Create an optional VPC and create 2 IAM users.
        UserNames userNames = createVpcUsers();
        String expressUserName = userNames.getExpressUserName();
        String regularUserName = userNames.getRegularUserName();

        //  Set up two S3 clients, one regular and one express,
        //  and two buckets, one regular and one directory.
        setupClientsAndBuckets(expressUserName, regularUserName);

        // Create an S3 session for the express S3 client and add objects to the
buckets.
        logger.info("Now let's add some objects to our buckets and demonstrate how
to work with S3 Sessions.");
        waitForInputToContinue(scanner);
        String bucketObject = createSessionAddObjects();

        // Demonstrate performance differences between regular and directory
buckets.
        demonstratePerformance(bucketObject);

        // Populate the buckets to show the lexicographical difference between
        // regular and express buckets.
        showLexicographicalDifferences(bucketObject);

        logger.info(DASHES);
        logger.info("That's it for our tour of the basic operations for S3 Express
One Zone.");
        logger.info("Would you like to cleanUp the AWS resources? (y/n): ");
        String response = scanner.next().trim().toLowerCase();
        if (response.equals("y")) {
            cleanUp(stackName);
        }
    }

    /*
```

```
     Delete resources created by this scenario.
   */
   public static void cleanUp(String stackName) {
       try {
           if (mdirectoryBucketName != null) {
               s3DirectoriesActions.deleteBucketAndObjectsAsync(mS3ExpressClient,
mdirectoryBucketName).join();
           }
           logger.info("Deleted directory bucket " + mdirectoryBucketName);
           mdirectoryBucketName = null;
           if (mregularBucketName != null) {
               s3DirectoriesActions.deleteBucketAndObjectsAsync(mS3RegularClient,
mregularBucketName).join();
           }
       } catch (CompletionException ce) {
           Throwable cause = ce.getCause();
           if (cause instanceof S3Exception) {
               logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
           } else {
               logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
           }
       }

       logger.info("Deleted regular bucket " + mregularBucketName);
       mregularBucketName = null;
       CloudFormationHelper.destroyCloudFormationStack(stackName);
   }

   private static void showLexicographicalDifferences(String bucketObject) {
       logger.info(DASHES);
       logger.info("""
           7. Populate the buckets to show the lexicographical (alphabetical)
difference
           when object names are listed. Now let's explore how directory buckets
store
           objects in a different manner to regular buckets. The key is in the
name
           "Directory". Where regular buckets store their key/value pairs in a
           flat manner, directory buckets use actual directories/folders.
           This allows for more rapid indexing, traversing, and therefore
           retrieval times!

           The more segmented your bucket is, with lots of
```

```
            directories, sub-directories, and objects, the more efficient it
becomes.
            This structural difference also causes `ListObject` operations to
behave
            differently, which can cause unexpected results. Let's add a few more
            objects in sub-directories to see how the output of
            ListObjects changes.
            """);

        waitForInputToContinue(scanner);

        //  Populate a few more files in each bucket so that we can use
        //  ListObjects and show the difference.
        String otherObject = "other/" + bucketObject;
        String altObject = "alt/" + bucketObject;
        String otherAltObject = "other/alt/" + bucketObject;

        try {
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, otherObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, otherObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, altObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, altObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, otherAltObject, "").join();
            s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, otherAltObject, "").join();

        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof NoSuchBucketException) {
                logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
            return;
        }

        try {
            // List objects in both S3 buckets.
```

```
            List<String> dirBucketObjects =
s3DirectoriesActions.listObjectsAsync(mS3ExpressClient,
mdirectoryBucketName).join();
            List<String> regBucketObjects =
s3DirectoriesActions.listObjectsAsync(mS3RegularClient, mregularBucketName).join();

            logger.info("Directory bucket content");
            for (String obj : dirBucketObjects) {
                logger.info(obj);
            }

            logger.info("Regular bucket content");
            for (String obj : regBucketObjects) {
                logger.info(obj);
            }
        } catch (CompletionException e) {
            logger.error("Async operation failed: {} ", e.getCause().getMessage());
            return;
        }

        logger.info("""
            Notice how the regular bucket lists objects in lexicographical order,
while the directory bucket does not. This is
            because the regular bucket considers the whole "key" to be the object
identifier, while the directory bucket actually
            creates directories and uses the object "key" as a path to the object.
            """);
        waitForInputToContinue(scanner);
    }

    /**
     * Demonstrates the performance difference between downloading an object from a
directory bucket and a regular bucket.
     *
     * <p>This method:
     * <ul>
     *     <li>Prompts the user to choose the number of downloads (default is
1,000).</li>
     *     <li>Downloads the specified object from the directory bucket and measures
the total time.</li>
     *     <li>Downloads the same object from the regular bucket and measures the
total time.</li>
     *     <li>Compares the time differences and prints the results.</li>
     * </ul>
```

```
    *
    * <p>Note: The performance difference will be more pronounced if this example
is run on an EC2 instance
    * in the same Availability Zone as the buckets.
    *
    * @param bucketObject the name of the object to download
    */
   private static void demonstratePerformance(String bucketObject) {
       logger.info(DASHES);
       logger.info("6. Demonstrate the performance difference.");
       logger.info("""
           Now, let's do a performance test. We'll download the same object from
each
           bucket repeatedly and compare the total time needed.

           Note: the performance difference will be much more pronounced if this
           example is run in an EC2 instance in the same Availability Zone as
           the bucket.
           """);
       waitForInputToContinue(scanner);

       int downloads = 1000; // Default value.
       logger.info("The default number of downloads of the same object for this
example is set at " + downloads + ".");

       // Ask if the user wants to download a different number.
       logger.info("Would you like to download the file a different number of
times? (y/n): ");
       String response = scanner.next().trim().toLowerCase();
       if (response.equals("y")) {
           int maxDownloads = 1_000_000;

           // Ask for a valid number of downloads.
           while (true) {
               logger.info("Enter a number between 1 and " + maxDownloads + " for
the number of downloads: ");
               if (scanner.hasNextInt()) {
                   downloads = scanner.nextInt();
                   if (downloads >= 1 && downloads <= maxDownloads) {
                       break;
                   } else {
                       logger.info("Please enter a number between 1 and " +
maxDownloads + ".");
                   }
```

```
                } else {
                    logger.info("Invalid input. Please enter a valid integer.");
                    scanner.next();
                }
            }

            logger.info("You have chosen to download {}  items.", downloads);
        } else {
            logger.info("No changes made. Using default downloads: {}", downloads);
        }
        // Simulating the download process for the directory bucket.
        logger.info("Downloading from the directory bucket.");
        long directoryTimeStart = System.nanoTime();
        for (int index = 0; index < downloads; index++) {
            if (index % 50 == 0) {
                logger.info("Download " + index + " of " + downloads);
            }

            try {
                // Get the object from the directory bucket.
                s3DirectoriesActions.getObjectAsync(mS3ExpressClient,
mdirectoryBucketName, bucketObject).join();
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof NoSuchKeyException) {
                    logger.error("S3Exception occurred: {}", cause.getMessage(),
ce);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                }
                return;
            }
        }
        long directoryTimeDifference = System.nanoTime() - directoryTimeStart;

        // Download from the regular bucket.
        logger.info("Downloading from the regular bucket.");
        long normalTimeStart = System.nanoTime();
        for (int index = 0; index < downloads; index++) {
            if (index % 50 == 0) {
                logger.info("Download " + index + " of " + downloads);
            }
```

```
            try {
                s3DirectoriesActions.getObjectAsync(mS3RegularClient,
mregularBucketName, bucketObject).join();
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof NoSuchKeyException) {
                    logger.error("S3Exception occurred: {}", cause.getMessage(),
ce);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                }
                return;
            }
        }

        long normalTimeDifference = System.nanoTime() - normalTimeStart;
        logger.info("The directory bucket took " + directoryTimeDifference
+ " nanoseconds, while the regular bucket took " + normalTimeDifference + "
nanoseconds.");
        long difference = normalTimeDifference - directoryTimeDifference;
        logger.info("That's a difference of " + difference + " nanoseconds, or");
        logger.info(difference / 1_000_000_000.0 + " seconds.");

        if (difference < 0) {
            logger.info("The directory buckets were slower. This can happen if you
are not running on the cloud within a VPC.");
        }
        waitForInputToContinue(scanner);
    }

    private static String createSessionAddObjects() {
        logger.info(DASHES);
        logger.info("""
            5. Create an object and copy it.
            We'll create an object consisting of some text and upload it to the
            regular bucket.
            """);
        waitForInputToContinue(scanner);

        String bucketObject = "basic-text-object.txt";
        try {
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, bucketObject, "Look Ma, I'm a bucket!").join();
```

```
            s3DirectoriesActions.createSessionAsync(mS3ExpressClient,
mdirectoryBucketName).join();

            // Copy the object to the destination S3 bucket.
            s3DirectoriesActions.copyObjectAsync(mS3ExpressClient,
mregularBucketName, bucketObject, mdirectoryBucketName, bucketObject).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof S3Exception) {
                logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
        }
        logger.info("""
            It worked! This is because the S3Client that performed the copy
operation
            is the expressClient using the credentials for the user with permission
to
            work with directory buckets.

            It's important to remember the user permissions when interacting with
            directory buckets. Instead of validating permissions on every call as
            regular buckets do, directory buckets utilize the user credentials and
session
            token to validate. This allows for much faster connection speeds on
every call.
            For single calls, this is low, but for many concurrent calls
            this adds up to a lot of time saved.
            """);
        waitForInputToContinue(scanner);
        return bucketObject;
    }

    /**
     * Creates VPC users for the S3 Express One Zone scenario.
     * <p>
     * This method performs the following steps:
     * <ol>
     *     <li>Optionally creates a new VPC and VPC Endpoint if the application
is running in an EC2 instance in the same Availability Zone as the directory
buckets.</li>
```

```
    *     <li>Creates two IAM users: one with S3 Express One Zone permissions and
one without.</li>
    * </ol>
    *
    * @return a {@link UserNames} object containing the names of the created IAM
users
    */
   public static UserNames createVpcUsers() {
       /*
       Optionally create a VPC.
       Create two IAM users, one with S3 Express One Zone permissions and one
without.
       */
       logger.info(DASHES);
       logger.info("""
           1. First, we'll set up a new VPC and VPC Endpoint if this program is
running in an EC2 instance in the same AZ as your\s
           directory buckets will be. Are you running this in an EC2 instance
located in the same AZ as your intended directory buckets?
           """);

       logger.info("Do you want to setup a VPC Endpoint? (y/n)");
       String endpointAns = scanner.nextLine().trim();
       if (endpointAns.equalsIgnoreCase("y")) {
           logger.info("""
               Great! Let's set up a VPC, retrieve the Route Table from it, and
create a VPC Endpoint to connect the S3 Client to.
               """);
           try {
               s3DirectoriesActions.setupVPCAsync().join();
           } catch (CompletionException ce) {
               Throwable cause = ce.getCause();
               if (cause instanceof Ec2Exception) {
                   logger.error("IamException occurred: {}", cause.getMessage(),
ce);
               } else {
                   logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
               }
           }
           waitForInputToContinue(scanner);
       } else {
           logger.info("Skipping the VPC setup. Don't forget to use this in
production!");
```

```
        }
        logger.info(DASHES);
        logger.info("""
            2. Create a RegularUser and ExpressUser by using the AWS CDK.
            One IAM User, named RegularUser, will have permissions to work only
            with regular buckets and one IAM user, named ExpressUser, will have
            permissions to work only with directory buckets.
            """);
        waitForInputToContinue(scanner);

        // Create two users required for this scenario.
        Map<String, String> stackOutputs = createUsersUsingCDK(stackName);
        regularUser = stackOutputs.get("RegularUser");
        expressUser = stackOutputs.get("ExpressUser");

        UserNames names = new UserNames();
        names.setRegularUserName(regularUser);
        names.setExpressUserName(expressUser);
        return names;
    }

    /**
     * Creates users using AWS CloudFormation.
     *
     * @return a {@link Map} of String keys and String values representing the stack
outputs,
     * which may include user-related information such as user names and IDs.
     */
    public static Map<String, String> createUsersUsingCDK(String stackName) {
        logger.info("We'll use an AWS CloudFormation template to create the IAM
users and policies.");
        CloudFormationHelper.deployCloudFormationStack(stackName);
        return CloudFormationHelper.getStackOutputsAsync(stackName).join();
    }

    /**
     * Sets up the necessary clients and buckets for the S3 Express service.
     *
     * @param expressUserName the username for the user with S3 Express permissions
     * @param regularUserName the username for the user with regular S3 permissions
     */
    public static void setupClientsAndBuckets(String expressUserName, String
regularUserName) {
        Scanner locscanner = new Scanner(System.in);
```

```
        String accessKeyIdforRegUser;
        String secretAccessforRegUser;
        try {
            CreateAccessKeyResponse keyResponse =
s3DirectoriesActions.createAccessKeyAsync(regularUserName).join();
            accessKeyIdforRegUser = keyResponse.accessKey().accessKeyId();
            secretAccessforRegUser = keyResponse.accessKey().secretAccessKey();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IamException) {
                logger.error("IamException occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
            return;
        }

        String accessKeyIdforExpressUser;
        String secretAccessforExpressUser;
        try {
            CreateAccessKeyResponse keyResponseExpress =
s3DirectoriesActions.createAccessKeyAsync(expressUserName).join();
            accessKeyIdforExpressUser =
keyResponseExpress.accessKey().accessKeyId();
            secretAccessforExpressUser =
keyResponseExpress.accessKey().secretAccessKey();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IamException) {
                logger.error("IamException occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
            return;
        }

        logger.info(DASHES);
        logger.info("""
            3. Create two S3Clients; one uses the ExpressUser's credentials and one
uses the RegularUser's credentials.
            The 2 S3Clients will use different credentials.
            """);
```

```
        waitForInputToContinue(locscanner);
        try {
            mS3RegularClient =
createS3ClientWithAccessKeyAsync(accessKeyIdforRegUser,
secretAccessforRegUser).join();
            mS3ExpressClient =
createS3ClientWithAccessKeyAsync(accessKeyIdforExpressUser,
secretAccessforExpressUser).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IllegalArgumentException) {
                logger.error("An invalid argument exception occurred: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
            return;
        }

        logger.info("""
            We can now use the ExpressUser client to make calls to S3 Express
operations.
            """);
        waitForInputToContinue(locscanner);
        logger.info(DASHES);
        logger.info("""
            4. Create two buckets.
            Now we will create a directory bucket which is the linchpin of the S3
Express One Zone service. Directory buckets
            behave differently from regular S3 buckets which we will explore here.
We'll also create a regular bucket, put
            an object into the regular bucket, and copy it to the directory bucket.
            """);

        logger.info("""
            Now, let's choose an availability zone (AZ) for the directory bucket.
            We'll choose one that is supported.
            """);
        String zoneId;
        String regularBucketName;
        try {
            zoneId = s3DirectoriesActions.selectAvailabilityZoneIdAsync().join();
            regularBucketName = "reg-bucket-" + System.currentTimeMillis();
```

```
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof Ec2Exception) {
                logger.error("EC2Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
            }
            return;
        }
        logger.info("""
            Now, let's create the actual directory bucket, as well as a regular
bucket."
             """);

        String directoryBucketName = "test-bucket-" + System.currentTimeMillis() +
"--" + zoneId + "--x-s3";
        try {
            s3DirectoriesActions.createDirectoryBucketAsync(mS3ExpressClient,
directoryBucketName, zoneId).join();
            logger.info("Created directory bucket {}", directoryBucketName);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof BucketAlreadyExistsException) {
                logger.error("The bucket already exists. Moving on: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
                return;
            }
        }

        // Assign to the data member.
        mdirectoryBucketName = directoryBucketName;
        try {
            s3DirectoriesActions.createBucketAsync(mS3RegularClient,
regularBucketName).join();
            logger.info("Created regular bucket {} ", regularBucketName);
            mregularBucketName = regularBucketName;
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof BucketAlreadyExistsException) {
```

```
                logger.error("The bucket already exists. Moving on: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}", cause.getMessage(),
ce);
                return;
            }
        }
        logger.info("Great! Both buckets were created.");
        waitForInputToContinue(locscanner);
    }

    /**
     * Creates an asynchronous S3 client with the specified access key and secret
access key.
     *
     * @param accessKeyId     the AWS access key ID
     * @param secretAccessKey the AWS secret access key
     * @return a {@link CompletableFuture} that asynchronously creates the S3 client
     * @throws IllegalArgumentException if the access key ID or secret access key is
null
     */
    public static CompletableFuture<S3AsyncClient>
createS3ClientWithAccessKeyAsync(String accessKeyId, String secretAccessKey) {
        return CompletableFuture.supplyAsync(() -> {
            // Validate input parameters
            if (accessKeyId == null || accessKeyId.isBlank() || secretAccessKey ==
null || secretAccessKey.isBlank()) {
                throw new IllegalArgumentException("Access Key ID and Secret Access
Key must not be null or empty");
            }

            AwsBasicCredentials awsCredentials =
AwsBasicCredentials.create(accessKeyId, secretAccessKey);
            return S3AsyncClient.builder()

.credentialsProvider(StaticCredentialsProvider.create(awsCredentials))
                .region(Region.US_WEST_2)
                .build();
        });
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
```

```
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}
```

Amazon S3 SDK 方法的包装器类。

```
public class S3DirectoriesActions {

    private static IamAsyncClient iamAsyncClient;

    private static Ec2AsyncClient ec2AsyncClient;
    private static final Logger logger =
 LoggerFactory.getLogger(S3DirectoriesActions.class);

    private static IamAsyncClient getIAMAsyncClient() {
        if (iamAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
 ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            iamAsyncClient = IamAsyncClient.builder()
```

```
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return iamAsyncClient;
    }

    private static Ec2AsyncClient getEc2AsyncClient() {
        if (ec2AsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            ec2AsyncClient = Ec2AsyncClient.builder()
                .httpClient(httpClient)
                .region(Region.US_WEST_2)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return ec2AsyncClient;
    }

    /**
     * Deletes the specified S3 bucket and all the objects within it asynchronously.
     *
     * @param s3AsyncClient the S3 asynchronous client to use for the operations
     * @param bucketName the name of the S3 bucket to be deleted
     * @return a {@link CompletableFuture} that completes with a {@link
WaiterResponse} containing the
     *         {@link HeadBucketResponse} when the bucket has been successfully
deleted
     * @throws CompletionException if there was an error deleting the bucket or its
objects
     */
```

```
   public CompletableFuture<WaiterResponse<HeadBucketResponse>>
deleteBucketAndObjectsAsync(S3AsyncClient s3AsyncClient, String bucketName) {
       ListObjectsV2Request listRequest = ListObjectsV2Request.builder()
           .bucket(bucketName)
           .build();

       return s3AsyncClient.listObjectsV2(listRequest)
           .thenCompose(listResponse -> {
               if (!listResponse.contents().isEmpty()) {
                   List<ObjectIdentifier> objectIdentifiers =
listResponse.contents().stream()
                       .map(s3Object ->
ObjectIdentifier.builder().key(s3Object.key()).build())
                       .collect(Collectors.toList());

                   DeleteObjectsRequest deleteRequest =
DeleteObjectsRequest.builder()
                       .bucket(bucketName)
                       .delete(Delete.builder().objects(objectIdentifiers).build())
                       .build();

                   return s3AsyncClient.deleteObjects(deleteRequest)
                       .thenAccept(deleteResponse -> {
                           if (!deleteResponse.errors().isEmpty()) {
                               deleteResponse.errors().forEach(error ->
                                   logger.error("Couldn't delete object " +
error.key() + ". Reason: " + error.message()));
                           }
                       });
               }
               return CompletableFuture.completedFuture(null);
           })
           .thenCompose(ignored -> {
               DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder()
                   .bucket(bucketName)
                   .build();
               return s3AsyncClient.deleteBucket(deleteBucketRequest);
           })
           .thenCompose(ignored -> {
               S3AsyncWaiter waiter = s3AsyncClient.waiter();
               HeadBucketRequest headBucketRequest =
HeadBucketRequest.builder().bucket(bucketName).build();
               return waiter.waitUntilBucketNotExists(headBucketRequest);
```

```
            })
            .whenComplete((ignored, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof S3Exception) {
                        throw new CompletionException("Error deleting bucket: " +
bucketName, cause);
                    }
                    throw new CompletionException("Failed to delete bucket and
objects: " + bucketName, exception);
                }
                logger.info("Bucket deleted successfully: " + bucketName);
            });
    }

    /**
     *  Lists the objects in an S3 bucket asynchronously.
     *
     * @param s3Client the S3 async client to use for the operation
     * @param bucketName the name of the S3 bucket containing the objects to list
     * @return a {@link CompletableFuture} that contains the list of object keys in
the specified bucket
     */
    public CompletableFuture<List<String>> listObjectsAsync(S3AsyncClient s3Client,
String bucketName) {
        ListObjectsV2Request request = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .build();

        return s3Client.listObjectsV2(request)
            .thenApply(response -> response.contents().stream()
                .map(S3Object::key)
                .toList())
            .whenComplete((result, exception) -> {
                if (exception != null) {
                    throw new CompletionException("Couldn't list objects in bucket:
" + bucketName, exception);
                }
            });
    }

    /**
     * Retrieves an object from an Amazon S3 bucket asynchronously.
     *
```

```
    * @param s3Client   the S3 async client to use for the operation
    * @param bucketName the name of the S3 bucket containing the object
    * @param keyName    the unique identifier (key) of the object to retrieve
    * @return a {@link CompletableFuture} that, when completed, contains the
object's content as a {@link ResponseBytes} of {@link GetObjectResponse}
    */
   public CompletableFuture<ResponseBytes<GetObjectResponse>>
getObjectAsync(S3AsyncClient s3Client, String bucketName, String keyName) {
       GetObjectRequest objectRequest = GetObjectRequest.builder()
           .key(keyName)
           .bucket(bucketName)
           .build();

       // Get the object asynchronously and transform it into a byte array
       return s3Client.getObject(objectRequest, AsyncResponseTransformer.toBytes())
           .exceptionally(exception -> {
               Throwable cause = exception.getCause();
               if (cause instanceof NoSuchKeyException) {
                   throw new CompletionException("Failed to get the object. Reason:
" + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
               }
               throw new CompletionException("Failed to get the object",
exception);
           });
   }

   /**
    * Asynchronously copies an object from one S3 bucket to another.
    *
    * @param s3Client           the S3 async client to use for the copy operation
    * @param sourceBucket       the name of the source bucket
    * @param sourceKey          the key of the object to be copied in the source
bucket
    * @param destinationBucket  the name of the destination bucket
    * @param destinationKey     the key of the copied object in the destination
bucket
    * @return a {@link CompletableFuture} that completes when the copy operation is
finished
    */
   public CompletableFuture<Void> copyObjectAsync(S3AsyncClient s3Client, String
sourceBucket, String sourceKey, String destinationBucket, String destinationKey) {
       CopyObjectRequest copyRequest = CopyObjectRequest.builder()
           .sourceBucket(sourceBucket)
           .sourceKey(sourceKey)
```

```
            .destinationBucket(destinationBucket)
            .destinationKey(destinationKey)
            .build();

        return s3Client.copyObject(copyRequest)
            .thenRun(() -> logger.info("Copied object '" + sourceKey + "' from
bucket '" + sourceBucket + "' to bucket '" + destinationBucket + "'"))
            .whenComplete((ignored, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof S3Exception) {
                        throw new CompletionException("Couldn't copy object '" +
sourceKey + "' from bucket '" + sourceBucket + "' to bucket '" + destinationBucket
+ "'. Reason: " + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
                    }
                    throw new CompletionException("Failed to copy object",
exception);
                }
            });
    }

    /**
     * Asynchronously creates a session for the specified S3 bucket.
     *
     * @param s3Client   the S3 asynchronous client to use for creating the session
     * @param bucketName the name of the S3 bucket for which to create the session
     * @return a {@link CompletableFuture} that completes when the session is
created, or throws a {@link CompletionException} if an error occurs
     */
    public CompletableFuture<CreateSessionResponse> createSessionAsync(S3AsyncClient
s3Client, String bucketName) {
        CreateSessionRequest request = CreateSessionRequest.builder()
            .bucket(bucketName)
            .build();

        return s3Client.createSession(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof S3Exception) {
                        throw new CompletionException("Couldn't create the session.
Reason: " + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
                    }
```

```
                    throw new CompletionException("Unexpected error occurred while
creating session", exception);
                }
                logger.info("Created session for bucket: " + bucketName);
            });

    }

    /**
     * Creates a new S3 directory bucket in a specified Zone (For example, a
     * specified Availability Zone in this code example).
     *
     * @param s3Client   The asynchronous S3 client used to create the bucket
     * @param bucketName The name of the bucket to be created
     * @param zone       The Availability Zone where the bucket will be created
     * @throws CompletionException if there's an error creating the bucket
     */
    public CompletableFuture<CreateBucketResponse>
createDirectoryBucketAsync(S3AsyncClient s3Client, String bucketName, String zone)
{
        logger.info("Creating bucket: " + bucketName);

        CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
            .location(LocationInfo.builder()
                .type(LocationType.AVAILABILITY_ZONE)
                .name(zone)
                .build())
            .bucket(BucketInfo.builder()
                .type(BucketType.DIRECTORY)
                .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                .build())
            .build();

        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .createBucketConfiguration(bucketConfiguration)
            .build();

        return s3Client.createBucket(bucketRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof BucketAlreadyExistsException) {
```

```
                        throw new CompletionException("The bucket already exists: "
+ ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
                    }
                    throw new CompletionException("Unexpected error occurred while
creating bucket", exception);
                }
                logger.info("Bucket created successfully with location: " +
response.location());
            });
    }

    /**
     * Creates an S3 bucket asynchronously.
     *
     * @param s3Client    the S3 async client to use for the bucket creation
     * @param bucketName  the name of the S3 bucket to create
     * @return a {@link CompletableFuture} that completes with the {@link
WaiterResponse} containing the {@link HeadBucketResponse}
     *         when the bucket is successfully created
     * @throws CompletionException if there's an error creating the bucket
     */
    public CompletableFuture<WaiterResponse<HeadBucketResponse>>
createBucketAsync(S3AsyncClient s3Client, String bucketName) {
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        return s3Client.createBucket(bucketRequest)
            .thenCompose(response -> {
                S3AsyncWaiter s3Waiter = s3Client.waiter();
                HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();
                return s3Waiter.waitUntilBucketExists(bucketRequestWait);
            })
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof BucketAlreadyExistsException) {
                        throw new CompletionException("The S3 bucket exists: " +
cause.getMessage(), cause);
                    } else {
                        throw new CompletionException("Failed to create access key:
" + exception.getMessage(), exception);
```

```
                    }
                }
                logger.info(bucketName + " is ready");
            });
    }

    /**
     * Uploads an object to an Amazon S3 bucket asynchronously.
     *
     * @param s3Client     the S3 async client to use for the upload
     * @param bucketName   the destination S3 bucket name
     * @param bucketObject the name of the object to be uploaded
     * @param text         the content to be uploaded as the object
     */
    public CompletableFuture<PutObjectResponse> putObjectAsync(S3AsyncClient
s3Client, String bucketName, String bucketObject, String text) {
        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(bucketObject)
            .build();

        return s3Client.putObject(objectRequest, AsyncRequestBody.fromString(text))
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof NoSuchBucketException) {
                        throw new CompletionException("The S3 bucket does not exist:
" + cause.getMessage(), cause);
                    } else {
                        throw new CompletionException("Failed to create access key:
" + exception.getMessage(), exception);
                    }
                }
            });
    }

    /**
     * Creates an AWS IAM access key asynchronously for the specified user name.
     *
     * @param userName the name of the IAM user for whom to create the access key
     * @return a {@link CompletableFuture} that completes with the {@link
CreateAccessKeyResponse} containing the created access key
     */
```

```
   public CompletableFuture<CreateAccessKeyResponse> createAccessKeyAsync(String
userName) {
       CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
           .userName(userName)
           .build();

       return getIAMAsyncClient().createAccessKey(request)
           .whenComplete((response, exception) -> {
               if (response != null) {
                   logger.info("Access Key Created.");
               } else {
                   if (exception == null) {
                       Throwable cause = exception.getCause();
                       if (cause instanceof IamException) {
                           throw new CompletionException("IAM error while creating
access key: " + cause.getMessage(), cause);
                       } else {
                           throw new CompletionException("Failed to create access
key: " + exception.getMessage(), exception);
                       }
                   }
               }
           });
   }

   /**
    * Asynchronously selects an Availability Zone ID from the available EC2 zones.
    *
    * @return A {@link CompletableFuture} that resolves to the selected
Availability Zone ID.
    * @throws CompletionException if an error occurs during the request or
processing.
    */
   public CompletableFuture<String> selectAvailabilityZoneIdAsync() {
       DescribeAvailabilityZonesRequest zonesRequest =
DescribeAvailabilityZonesRequest.builder()
           .build();

       return getEc2AsyncClient().describeAvailabilityZones(zonesRequest)
           .thenCompose(response -> {
               List<AvailabilityZone> zonesList = response.availabilityZones();
               if (zonesList.isEmpty()) {
                   logger.info("No availability zones found.");
```

```
                    return CompletableFuture.completedFuture(null); // Return null
 if no zones are found
                }

                List<String> zoneIds = zonesList.stream()
                    .map(AvailabilityZone::zoneId) // Get the zoneId (e.g., "usw2-
az1")
                    .toList();

                return CompletableFuture.supplyAsync(() ->
 promptUserForZoneSelection(zonesList, zoneIds))
                    .thenApply(selectedZone -> {
                        // Return only the selected Zone ID (e.g., "usw2-az1").
                        return selectedZone.zoneId();
                    });
            })
            .whenComplete((result, exception) -> {
                if (exception == null) {
                    if (result != null) {
                        logger.info("Selected Availability Zone ID: " + result);
                    } else {
                        logger.info("No availability zone selected.");
                    }
                } else {
                    Throwable cause = exception.getCause();
                    if (cause instanceof Ec2Exception) {
                        throw new CompletionException("EC2 error while selecting
 availability zone: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Failed to select availability
 zone: " + exception.getMessage(), exception);
                }
            });
    }

    /**
     * Prompts the user to select an Availability Zone from the given list.
     *
     * @param zonesList the list of Availability Zones
     * @param zoneIds the list of zone IDs
     * @return the selected Availability Zone
     */
```

```
   private static AvailabilityZone
promptUserForZoneSelection(List<AvailabilityZone> zonesList, List<String> zoneIds)
{
       Scanner scanner = new Scanner(System.in);
       int index = -1;

       while (index < 0 || index >= zoneIds.size()) {
           logger.info("Select an availability zone:");
           IntStream.range(0, zoneIds.size()).forEach(i ->
               logger.info(i + ": " + zoneIds.get(i))
           );

           logger.info("Enter the number corresponding to your choice: ");
           if (scanner.hasNextInt()) {
               index = scanner.nextInt();
           } else {
               scanner.next();
           }
       }

       AvailabilityZone selectedZone = zonesList.get(index);
       logger.info("You selected: " + selectedZone.zoneId());
       return selectedZone;
   }

   /**
    * Asynchronously sets up a new VPC, including creating the VPC, finding the
associated route table, and
    * creating a VPC endpoint for the S3 service.
    *
    * @return a {@link CompletableFuture} that, when completed, contains a
AbstractMap with the
    *         VPC ID and VPC endpoint ID.
    */
   public CompletableFuture<AbstractMap.SimpleEntry<String, String>>
setupVPCAsync() {
       String cidr = "10.0.0.0/16";
       CreateVpcRequest vpcRequest = CreateVpcRequest.builder()
           .cidrBlock(cidr)
           .build();

       return getEc2AsyncClient().createVpc(vpcRequest)
           .thenCompose(vpcResponse -> {
               String vpcId = vpcResponse.vpc().vpcId();
```

```
                logger.info("VPC Created: {}", vpcId);

                Ec2AsyncWaiter waiter = getEc2AsyncClient().waiter();
                DescribeVpcsRequest request = DescribeVpcsRequest.builder()
                    .vpcIds(vpcId)
                    .build();

                return waiter.waitUntilVpcAvailable(request)
                    .thenApply(waiterResponse -> vpcId);
            })
            .thenCompose(vpcId -> {
                Filter filter = Filter.builder()
                    .name("vpc-id")
                    .values(vpcId)
                    .build();

                DescribeRouteTablesRequest describeRouteTablesRequest =
DescribeRouteTablesRequest.builder()
                    .filters(filter)
                    .build();

                return
getEc2AsyncClient().describeRouteTables(describeRouteTablesRequest)
                    .thenApply(routeTablesResponse -> {
                        if (routeTablesResponse.routeTables().isEmpty()) {
                            throw new CompletionException("No route tables found for
VPC: " + vpcId, null);
                        }
                        String routeTableId =
routeTablesResponse.routeTables().get(0).routeTableId();
                        logger.info("Route table found: {}", routeTableId);
                        return new AbstractMap.SimpleEntry<>(vpcId, routeTableId);
                    });
            })
            .thenCompose(vpcAndRouteTable -> {
                String vpcId = vpcAndRouteTable.getKey();
                String routeTableId = vpcAndRouteTable.getValue();
                Region region =
getEc2AsyncClient().serviceClientConfiguration().region();
                String serviceName = String.format("com.amazonaws.%s.s3express",
region.id());

                CreateVpcEndpointRequest endpointRequest =
CreateVpcEndpointRequest.builder()
```

```
                    .vpcId(vpcId)
                    .routeTableIds(routeTableId)
                    .serviceName(serviceName)
                    .build();

                return getEc2AsyncClient().createVpcEndpoint(endpointRequest)
                    .thenApply(vpcEndpointResponse -> {
                        String vpcEndpointId =
 vpcEndpointResponse.vpcEndpoint().vpcEndpointId();
                        logger.info("VPC Endpoint created: {}", vpcEndpointId);
                        return new AbstractMap.SimpleEntry<>(vpcId, vpcEndpointId);
                    });
            })
            .exceptionally(exception -> {
                Throwable cause = exception.getCause() != null ?
 exception.getCause() : exception;
                if (cause instanceof Ec2Exception) {
                    logger.error("EC2 error during VPC setup: {}",
 cause.getMessage(), cause);
                    throw new CompletionException("EC2 error during VPC setup: " +
 cause.getMessage(), cause);
                }

                logger.error("VPC setup failed: {}", cause.getMessage(), cause);
                throw new CompletionException("VPC setup failed: " +
 cause.getMessage(), cause);
            });
    }

}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObject
  - GetObject
  - ListObjects
  - PutObject

操作

AbortMultipartUpload

以下代码示例演示了如何使用 AbortMultipartUpload。

适用于 Java 的 SDK 2.x

> **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。

中止目录存储桶中的分段上传。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;


    /**
     * Aborts a specific multipart upload for the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be uploaded
     * @param uploadId   The upload ID of the multipart upload to abort
     * @return True if the multipart upload is successfully aborted, false otherwise
     */
    public static boolean abortDirectoryBucketMultipartUpload(S3Client s3Client,
 String bucketName,
            String objectKey, String uploadId) {
```

```
        logger.info("Aborting multipart upload: {} for bucket: {}", uploadId,
bucketName);
        try {
            // Abort the multipart upload
            AbortMultipartUploadRequest abortMultipartUploadRequest =
AbortMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .uploadId(uploadId)
                    .build();

            s3Client.abortMultipartUpload(abortMultipartUploadRequest);
            logger.info("Aborted multipart upload: {} for object: {}", uploadId,
objectKey);
            return true;
        } catch (S3Exception e) {
            logger.error("Failed to abort multipart upload: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考AbortMultipartUpload中的。

`CompleteMultipartUpload`

以下代码示例演示了如何使用 `CompleteMultipartUpload`。

适用于 Java 的 SDK 2.x

> **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。

完成目录存储桶中的分段上传。

```
import com.example.s3.util.S3DirectoryBucketUtils;
```

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;


    /**
     * This method completes the multipart upload request by collating all the
     * upload parts.
     *
     * @param s3Client    The S3 client used to interact with S3
     * @param bucketName  The name of the directory bucket
     * @param objectKey   The key (name) of the object to be uploaded
     * @param uploadId    The upload ID used to track the multipart upload
     * @param uploadParts The list of completed parts
     * @return True if the multipart upload is successfully completed, false
     *         otherwise
     */
    public static boolean completeDirectoryBucketMultipartUpload(S3Client s3Client,
 String bucketName, String objectKey,
            String uploadId, List<CompletedPart> uploadParts) {
        try {
            CompletedMultipartUpload completedMultipartUpload =
 CompletedMultipartUpload.builder()
```

```
                    .parts(uploadParts)
                    .build();
            CompleteMultipartUploadRequest completeMultipartUploadRequest =
CompleteMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .uploadId(uploadId)
                    .multipartUpload(completedMultipartUpload)
                    .build();

            CompleteMultipartUploadResponse response =
s3Client.completeMultipartUpload(completeMultipartUploadRequest);
            logger.info("Multipart upload completed. ETag: {}", response.eTag());
            return true;
        } catch (S3Exception e) {
            logger.error("Failed to complete multipart upload: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CompleteMultipartUpload中的。

`CopyObject`

以下代码示例演示了如何使用 `CopyObject`。

适用于 Java 的 SDK 2.x

> **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。

将对象从目录存储桶复制到目录存储桶。

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

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Copies an object from one S3 general purpose bucket to one S3 directory
     * bucket.
     *
     * @param s3Client     The S3 client used to interact with S3
     * @param sourceBucket The name of the source bucket
     * @param objectKey    The key (name) of the object to be copied
     * @param targetBucket The name of the target bucket
     */
    public static void copyDirectoryBucketObject(S3Client s3Client, String
 sourceBucket, String objectKey,
            String targetBucket) {
        logger.info("Copying object: {} from bucket: {} to bucket: {}", objectKey,
 sourceBucket, targetBucket);

        try {
            // Create a CopyObjectRequest
            CopyObjectRequest copyReq = CopyObjectRequest.builder()
                    .sourceBucket(sourceBucket)
                    .sourceKey(objectKey)
                    .destinationBucket(targetBucket)
                    .destinationKey(objectKey)
                    .build();

            // Copy the object
            CopyObjectResponse copyRes = s3Client.copyObject(copyReq);
            logger.info("Successfully copied {} from bucket {} into bucket {}.
 CopyObjectResponse: {}",
```

```
                    objectKey, sourceBucket, targetBucket,
copyRes.copyObjectResult().toString());

        } catch (S3Exception e) {
            logger.error("Failed to copy object: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CopyObject中的。

`CreateBucket`

以下代码示例演示了如何使用 `CreateBucket`。

适用于 Java 的 SDK 2.x

> **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。

创建 S3 目录存储桶。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketInfo;
import software.amazon.awssdk.services.s3.model.BucketType;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateBucketResponse;
import software.amazon.awssdk.services.s3.model.DataRedundancy;
import software.amazon.awssdk.services.s3.model.LocationInfo;
import software.amazon.awssdk.services.s3.model.LocationType;
import software.amazon.awssdk.services.s3.model.S3Exception;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * Creates a new S3 directory bucket in a specified Zone (For example, a
     * specified Availability Zone in this code example).
     *
     * @param s3Client   The S3 client used to create the bucket
     * @param bucketName The name of the bucket to be created
     * @param zone       The region where the bucket will be created
     * @throws S3Exception if there's an error creating the bucket
     */
    public static void createDirectoryBucket(S3Client s3Client, String bucketName,
 String zone) throws S3Exception {
        logger.info("Creating bucket: {}", bucketName);

        CreateBucketConfiguration bucketConfiguration =
 CreateBucketConfiguration.builder()
                .location(LocationInfo.builder()
                        .type(LocationType.AVAILABILITY_ZONE)
                        .name(zone).build())
                .bucket(BucketInfo.builder()
                        .type(BucketType.DIRECTORY)
                        .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                        .build())
                .build();
        try {
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .createBucketConfiguration(bucketConfiguration).build();
            CreateBucketResponse response = s3Client.createBucket(bucketRequest);
            logger.info("Bucket created successfully with location: {}",
 response.location());
        } catch (S3Exception e) {
            logger.error("Error creating bucket: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateBucket中的。

`CreateMultipartUpload`

以下代码示例演示了如何使用 `CreateMultipartUpload`。

适用于 Java 的 SDK 2.x

> **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。

在目录存储桶中创建分段上传。

```
import com.example.s3.util.S3DirectoryBucketUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * This method creates a multipart upload request that generates a unique upload
     * ID used to track
     * all the upload parts.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be uploaded
     * @return The upload ID used to track the multipart upload
     */
    public static String createDirectoryBucketMultipartUpload(S3Client s3Client,
 String bucketName, String objectKey) {
        logger.info("Creating multipart upload for object: {} in bucket: {}",
 objectKey, bucketName);

        try {
```

```
            // Create a CreateMultipartUploadRequest
            CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Initiate the multipart upload
            CreateMultipartUploadResponse response =
s3Client.createMultipartUpload(createMultipartUploadRequest);
            String uploadId = response.uploadId();
            logger.info("Multipart upload initiated. Upload ID: {}", uploadId);
            return uploadId;

        } catch (S3Exception e) {
            logger.error("Failed to create multipart upload: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateMultipartUpload中的。

`DeleteBucket`

以下代码示例演示了如何使用 `DeleteBucket`。

适用于 Java 的 SDK 2.x

> **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。

删除 S3 目录存储桶。

```
import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;

    /**
     * Deletes the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket to delete
     */
    public static void deleteDirectoryBucket(S3Client s3Client, String bucketName) {
        logger.info("Deleting bucket: {}", bucketName);

        try {
            // Create a DeleteBucketRequest
            DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Delete the bucket
            s3Client.deleteBucket(deleteBucketRequest);
            logger.info("Successfully deleted bucket: {}", bucketName);

        } catch (S3Exception e) {
            logger.error("Failed to delete bucket: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteBucket中的。

DeleteBucketEncryption

以下代码示例演示了如何使用 DeleteBucketEncryption。

适用于 Java 的 SDK 2.x

> **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。

删除目录存储桶的加密配置。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * Deletes the encryption configuration from an S3 bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     */
    public static void deleteDirectoryBucketEncryption(S3Client s3Client, String
 bucketName) {
        DeleteBucketEncryptionRequest deleteRequest =
 DeleteBucketEncryptionRequest.builder()
                .bucket(bucketName)
                .build();

        try {
            s3Client.deleteBucketEncryption(deleteRequest);
            logger.info("Bucket encryption deleted for bucket: {}", bucketName);
        } catch (S3Exception e) {
            logger.error("Failed to delete bucket encryption: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
```

```
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteBucketEncryption中的。

`DeleteBucketPolicy`

以下代码示例演示了如何使用 `DeleteBucketPolicy`。

适用于 Java 的 SDK 2.x

> **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。

删除目录存储桶的存储桶策略。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketPolicy;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;


    /**
     * Deletes the bucket policy for the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     */
```

```
    public static void deleteDirectoryBucketPolicy(S3Client s3Client, String
bucketName) {
        logger.info("Deleting policy for bucket: {}", bucketName);

        try {
            // Create a DeleteBucketPolicyRequest
            DeleteBucketPolicyRequest deletePolicyReq =
DeleteBucketPolicyRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Delete the bucket policy
            s3Client.deleteBucketPolicy(deletePolicyReq);
            logger.info("Successfully deleted bucket policy");

        } catch (S3Exception e) {
            logger.error("Failed to delete bucket policy: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteBucketPolicy中的。

`DeleteObject`

以下代码示例演示了如何使用 `DeleteObject`。

适用于 Java 的 SDK 2.x

> **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。

删除目录存储桶中的一个对象。

```
import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;




    /**
     * Deletes an object from the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be deleted
     */
    public static void deleteDirectoryBucketObject(S3Client s3Client, String
 bucketName, String objectKey) {
        logger.info("Deleting object: {} from bucket: {}", objectKey, bucketName);

        try {
            // Create a DeleteObjectRequest
            DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Delete the object
            s3Client.deleteObject(deleteObjectRequest);
            logger.info("Object {} has been deleted", objectKey);

        } catch (S3Exception e) {
            logger.error("Failed to delete object: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
```

```
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteObject中的。

`DeleteObjects`

以下代码示例演示了如何使用 `DeleteObjects`。

适用于 Java 的 SDK 2.x

> **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。

删除目录存储桶中的多个对象。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectsResponse;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.net.URISyntaxException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;
```

```
   /**
    * Deletes multiple objects from the specified S3 directory bucket.
    *
    * @param s3Client   The S3 client used to interact with S3
    * @param bucketName The name of the directory bucket
    * @param objectKeys The list of keys (names) of the objects to be deleted
    */
   public static void deleteDirectoryBucketObjects(S3Client s3Client, String
bucketName, List<String> objectKeys) {
       logger.info("Deleting objects from bucket: {}", bucketName);

       try {
           // Create a list of ObjectIdentifier.
           List<ObjectIdentifier> identifiers = objectKeys.stream()
                   .map(key -> ObjectIdentifier.builder().key(key).build())
                   .toList();

           Delete delete = Delete.builder()
                   .objects(identifiers)
                   .build();

           DeleteObjectsRequest deleteObjectsRequest =
DeleteObjectsRequest.builder()
                   .bucket(bucketName)
                   .delete(delete)
                   .build();

           DeleteObjectsResponse deleteObjectsResponse =
s3Client.deleteObjects(deleteObjectsRequest);
           deleteObjectsResponse.deleted().forEach(deleted -> logger.info("Deleted
object: {}", deleted.key()));

       } catch (S3Exception e) {
           logger.error("Failed to delete objects: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                   e.awsErrorDetails().errorCode(), e);
           throw e;
       }
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteObjects中的。

GetBucketEncryption

以下代码示例演示了如何使用 GetBucketEncryption。

适用于 Java 的 SDK 2.x

> **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。

获取目录存储桶的加密配置。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.GetBucketEncryptionResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionRule;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * Retrieves the encryption configuration for an S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @return The type of server-side encryption applied to the bucket (e.g.,
     *         AES256, aws:kms)
     */
    public static String getDirectoryBucketEncryption(S3Client s3Client, String
 bucketName) {
        try {
            // Create a GetBucketEncryptionRequest
            GetBucketEncryptionRequest getRequest =
 GetBucketEncryptionRequest.builder()
                    .bucket(bucketName)
```

```
                    .build();

            // Retrieve the bucket encryption configuration
            GetBucketEncryptionResponse response =
s3Client.getBucketEncryption(getRequest);
            ServerSideEncryptionRule rule =
response.serverSideEncryptionConfiguration().rules().get(0);

            String encryptionType =
rule.applyServerSideEncryptionByDefault().sseAlgorithmAsString();
            logger.info("Bucket encryption algorithm: {}", encryptionType);
            logger.info("KMS Customer Managed Key ID: {}",
rule.applyServerSideEncryptionByDefault().kmsMasterKeyID());
            logger.info("Bucket Key Enabled: {}", rule.bucketKeyEnabled());

            return encryptionType;
        } catch (S3Exception e) {
            logger.error("Failed to get bucket encryption: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetBucketEncryption中的。

`GetBucketPolicy`

以下代码示例演示了如何使用 `GetBucketPolicy`。

适用于 Java 的 SDK 2.x

> **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。

获取目录存储桶的策略。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketPolicy;

    /**
     * Retrieves the bucket policy for the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @return The bucket policy text
     */
    public static String getDirectoryBucketPolicy(S3Client s3Client, String
 bucketName) {
        logger.info("Getting policy for bucket: {}", bucketName);

        try {
            // Create a GetBucketPolicyRequest
            GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Retrieve the bucket policy
            GetBucketPolicyResponse response = s3Client.getBucketPolicy(policyReq);

            // Print and return the policy text
            String policyText = response.policy();
            logger.info("Bucket policy: {}", policyText);
            return policyText;

        } catch (S3Exception e) {
            logger.error("Failed to get bucket policy: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
```

```
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetBucketPolicy中的。

`GetObject`

以下代码示例演示了如何使用 `GetObject`。

适用于 Java 的 SDK 2.x

> **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。

获取目录存储桶中的一个对象。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.charset.StandardCharsets;
import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
```

```
     * Retrieves an object from the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be retrieved
     * @return The retrieved object as a ResponseInputStream
     */
    public static boolean getDirectoryBucketObject(S3Client s3Client, String
bucketName, String objectKey) {
        logger.info("Retrieving object: {} from bucket: {}", objectKey, bucketName);

        try {
            // Create a GetObjectRequest
            GetObjectRequest objectRequest = GetObjectRequest.builder()
                    .key(objectKey)
                    .bucket(bucketName)
                    .build();

            // Retrieve the object as bytes
            ResponseBytes<GetObjectResponse> objectBytes =
s3Client.getObjectAsBytes(objectRequest);
            byte[] data = objectBytes.asByteArray();

            // Print object contents to console
            String objectContent = new String(data, StandardCharsets.UTF_8);
            logger.info("Object contents: \n{}", objectContent);

            return true;

        } catch (S3Exception e) {
            logger.error("Failed to retrieve object: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetObject中的。

GetObjectAttributes

以下代码示例演示了如何使用 GetObjectAttributes。

适用于 Java 的 SDK 2.x

> **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。

获取目录存储桶中的对象属性。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAttributesRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAttributesResponse;
import software.amazon.awssdk.services.s3.model.ObjectAttributes;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Retrieves attributes for an object in the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to retrieve attributes for
     * @return True if the object attributes are successfully retrieved, false
     *         otherwise
     */
    public static boolean getDirectoryBucketObjectAttributes(S3Client s3Client,
 String bucketName, String objectKey) {
        logger.info("Retrieving attributes for object: {} from bucket: {}",
 objectKey, bucketName);
```

```
        try {
            // Create a GetObjectAttributesRequest
            GetObjectAttributesRequest getObjectAttributesRequest =
GetObjectAttributesRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .objectAttributes(ObjectAttributes.E_TAG,
ObjectAttributes.STORAGE_CLASS,
                            ObjectAttributes.OBJECT_SIZE)
                    .build();

            // Retrieve the object attributes
            GetObjectAttributesResponse response =
s3Client.getObjectAttributes(getObjectAttributesRequest);
            logger.info("Attributes for object {}:", objectKey);
            logger.info("ETag: {}", response.eTag());
            logger.info("Storage Class: {}", response.storageClass());
            logger.info("Object Size: {}", response.objectSize());
            return true;

        } catch (S3Exception e) {
            logger.error("Failed to retrieve object attributes: {} - Error code:
{}",
                    e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[GetObjectAttributes](#)中的。

HeadBucket

以下代码示例演示了如何使用 HeadBucket。

适用于 Java 的 SDK 2.x

> **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。

检查指定的 S3 目录存储桶是否存在并且可供访问。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

    /**
     * Checks if the specified S3 directory bucket exists and is accessible.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket to check
     * @return True if the bucket exists and is accessible, false otherwise
     */
    public static boolean headDirectoryBucket(S3Client s3Client, String bucketName)
 {
        logger.info("Checking if bucket exists: {}", bucketName);

        try {
            // Create a HeadBucketRequest
            HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();
            // If the bucket doesn't exist, the following statement throws
 NoSuchBucketException,
            // which is a subclass of S3Exception.
            s3Client.headBucket(headBucketRequest);
            logger.info("Amazon S3 directory bucket: \"{}\" found.", bucketName);
```

```
            return true;

        } catch (S3Exception e) {
            logger.error("Failed to access bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考HeadBucket中的。

`HeadObject`

以下代码示例演示了如何使用 `HeadObject`。

适用于 Java 的 SDK 2.x

> **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。

获取目录存储桶中对象的元数据。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Retrieves metadata for an object in the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to retrieve metadata for
     * @return True if the object exists, false otherwise
     */
    public static boolean headDirectoryBucketObject(S3Client s3Client, String
 bucketName, String objectKey) {
        logger.info("Retrieving metadata for object: {} from bucket: {}", objectKey,
 bucketName);

        try {
            // Create a HeadObjectRequest
            HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Retrieve the object metadata
            HeadObjectResponse response = s3Client.headObject(headObjectRequest);
            logger.info("Amazon S3 object: \"{}\" found in bucket: \"{}\" with ETag:
 \"{}\"", objectKey, bucketName,
                    response.eTag());
            logger.info("Content-Type: {}", response.contentType());
            logger.info("Content-Length: {}", response.contentLength());
            logger.info("Last Modified: {}", response.lastModified());
            return true;

        } catch (S3Exception e) {
            logger.error("Failed to retrieve object metadata: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            return false;
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[HeadObject](#)中的。

`ListDirectoryBuckets`

以下代码示例演示了如何使用 `ListDirectoryBuckets`。

适用于 Java 的 SDK 2.x

> **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。

列出所有目录存储桶。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.List;
import java.util.UUID;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

     /**
     * Lists all S3 directory buckets and no general purpose buckets.
     *
     * @param s3Client The S3 client used to interact with S3
     * @return A list of bucket names
     */
    public static List<String> listDirectoryBuckets(S3Client s3Client) {
        logger.info("Listing all directory buckets");

        try {
            // Create a ListBucketsRequest
```

```
            ListDirectoryBucketsRequest listDirectoryBucketsRequest =
ListDirectoryBucketsRequest.builder().build();

            // Retrieve the list of buckets
            ListDirectoryBucketsResponse response =
s3Client.listDirectoryBuckets(listDirectoryBucketsRequest);

            // Extract bucket names
            List<String> bucketNames = response.buckets().stream()
                    .map(Bucket::name)
                    .collect(Collectors.toList());

            return bucketNames;
        } catch (S3Exception e) {
            logger.error("Failed to list buckets: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
            throw e;
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListDirectoryBuckets中的。

`ListMultipartUploads`

以下代码示例演示了如何使用 `ListMultipartUploads`。

适用于 Java 的 SDK 2.x

> **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。

列出目录存储桶中的分段上传。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static
 com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;


   /**
    * Lists multipart uploads for the specified S3 directory bucket.
    *
    * @param s3Client   The S3 client used to interact with S3
    * @param bucketName The name of the directory bucket
    * @return A list of MultipartUpload objects representing the multipart uploads
    */
   public static List<MultipartUpload> listDirectoryBucketMultipartUploads(S3Client
 s3Client, String bucketName) {
       logger.info("Listing in-progress multipart uploads for bucket: {}",
 bucketName);

       try {
           // Create a ListMultipartUploadsRequest
           ListMultipartUploadsRequest listMultipartUploadsRequest =
 ListMultipartUploadsRequest.builder()
                   .bucket(bucketName)
                   .build();

           // List the multipart uploads
           ListMultipartUploadsResponse response =
 s3Client.listMultipartUploads(listMultipartUploadsRequest);
```

```
            List<MultipartUpload> uploads = response.uploads();
            for (MultipartUpload upload : uploads) {
                logger.info("In-progress multipart upload: Upload ID: {}, Key: {},
Initiated: {}", upload.uploadId(),
                        upload.key(), upload.initiated());
            }
            return uploads;

        } catch (S3Exception e) {
            logger.error("Failed to list multipart uploads: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
            return List.of(); // Return an empty list if an exception is thrown
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ListMultipartUploads](#)中的。

`ListObjectsV2`

以下代码示例演示了如何使用 `ListObjectsV2`。

适用于 Java 的 SDK 2.x

> **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。

列出目录存储桶中的对象。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;
```

```
import java.nio.file.Path;
import java.util.List;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

    /**
     * Lists objects in the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @return A list of object keys in the bucket
     */
    public static List<String> listDirectoryBucketObjectsV2(S3Client s3Client,
 String bucketName) {
        logger.info("Listing objects in bucket: {}", bucketName);

        try {
            // Create a ListObjectsV2Request
            ListObjectsV2Request listObjectsV2Request =
 ListObjectsV2Request.builder()
                    .bucket(bucketName)
                    .build();

            // Retrieve the list of objects
            ListObjectsV2Response response =
 s3Client.listObjectsV2(listObjectsV2Request);

            // Extract and return the object keys
            return response.contents().stream()
                    .map(S3Object::key)
                    .collect(Collectors.toList());

        } catch (S3Exception e) {
            logger.error("Failed to list objects: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
```

```
            throw e;
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考中的 ListObjectsV2。

`ListParts`

以下代码示例演示了如何使用 `ListParts`。

适用于 Java 的 SDK 2.x

> **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。

列出目录存储桶中分段上传的分段。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListPartsRequest;
import software.amazon.awssdk.services.s3.model.ListPartsResponse;
import software.amazon.awssdk.services.s3.model.Part;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static
 com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

   /**
    * Lists the parts of a multipart upload for the specified S3 directory bucket.
    *
    * @param s3Client   The S3 client used to interact with S3
    * @param bucketName The name of the directory bucket
    * @param objectKey  The key (name) of the object being uploaded
    * @param uploadId   The upload ID used to track the multipart upload
    * @return A list of Part representing the parts of the multipart upload
    */
   public static List<Part> listDirectoryBucketMultipartUploadParts(S3Client
 s3Client, String bucketName,
           String objectKey, String uploadId) {
       logger.info("Listing parts for object: {} in bucket: {}", objectKey,
 bucketName);

       try {
           // Create a ListPartsRequest
           ListPartsRequest listPartsRequest = ListPartsRequest.builder()
                   .bucket(bucketName)
                   .uploadId(uploadId)
                   .key(objectKey)
                   .build();

           // List the parts of the multipart upload
           ListPartsResponse response = s3Client.listParts(listPartsRequest);
           List<Part> parts = response.parts();
           for (Part part : parts) {
               logger.info("Uploaded part: Part number = \"{}\", etag = {}",
 part.partNumber(), part.eTag());
           }
           return parts;

       } catch (S3Exception e) {
           logger.error("Failed to list parts: {} - Error code: {}",
 e.awsErrorDetails().errorMessage(),
                   e.awsErrorDetails().errorCode());
           return List.of(); // Return an empty list if an exception is thrown
       }
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListParts中的。

PutBucketEncryption

以下代码示例演示了如何使用 PutBucketEncryption。

适用于 Java 的 SDK 2.x

> **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。

为目录存储桶设置存储桶加密。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.ServerSideEncryption;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionByDefault;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionConfiguration;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionRule;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createKmsClient;
import static com.example.s3.util.S3DirectoryBucketUtils.createKmsKey;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.scheduleKeyDeletion;

    /**
     * Sets the default encryption configuration for an S3 bucket as SSE-KMS.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
```

```
     * @param kmsKeyId   The ID of the customer-managed KMS key
     */
    public static void putDirectoryBucketEncryption(S3Client s3Client, String
bucketName, String kmsKeyId) {
        // Define the default encryption configuration to use SSE-KMS. For directory
        // buckets, AWS managed KMS keys aren't supported. Only customer-managed
keys
        // are supported.
        ServerSideEncryptionByDefault encryptionByDefault =
ServerSideEncryptionByDefault.builder()
                .sseAlgorithm(ServerSideEncryption.AWS_KMS)
                .kmsMasterKeyID(kmsKeyId)
                .build();

        // Create a server-side encryption rule to apply the default encryption
        // configuration. For directory buckets, the bucketKeyEnabled field is
enforced
        // to be true.
        ServerSideEncryptionRule rule = ServerSideEncryptionRule.builder()
                .bucketKeyEnabled(true)
                .applyServerSideEncryptionByDefault(encryptionByDefault)
                .build();

        // Create the server-side encryption configuration for the bucket
        ServerSideEncryptionConfiguration encryptionConfiguration =
ServerSideEncryptionConfiguration.builder()
                .rules(rule)
                .build();

        // Create the PutBucketEncryption request
        PutBucketEncryptionRequest putRequest = PutBucketEncryptionRequest.builder()
                .bucket(bucketName)
                .serverSideEncryptionConfiguration(encryptionConfiguration)
                .build();

        // Set the bucket encryption
        try {
            s3Client.putBucketEncryption(putRequest);
            logger.info("SSE-KMS Bucket encryption configuration set for the
directory bucket: {}", bucketName);
        } catch (S3Exception e) {
            logger.error("Failed to set bucket encryption: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
```

```
            throw e;
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutBucketEncryption中的。

`PutBucketPolicy`

以下代码示例演示了如何使用 PutBucketPolicy。

适用于 Java 的 SDK 2.x

> **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。

将存储桶策略应用于目录存储桶。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;

    /**
     * Sets the following bucket policy for the specified S3 directory bucket.
     *<pre>
     * {
     *     "Version":"2012-10-17",
     *     "Statement": [
     *         {
```

```
     *             "Sid": "AdminPolicy",
     *             "Effect": "Allow",
     *             "Principal": {
     *                 "AWS": "arn:aws:iam::<ACCOUNT_ID>:root"
     *             },
     *             "Action": "s3express:*",
     *             "Resource": "arn:aws:s3express:us-west-2:<ACCOUNT_ID>:bucket/
<DIR_BUCKET_NAME>
     *         }
     *     ]
     * }
     * </pre>
     * This policy grants all S3 directory bucket actions to identities in the same
 account as the bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param policyText The policy text to be applied
     */
    public static void putDirectoryBucketPolicy(S3Client s3Client, String
 bucketName, String policyText) {
        logger.info("Setting policy on bucket: {}", bucketName);
        logger.info("Policy: {}", policyText);

        try {
            PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
                    .bucket(bucketName)
                    .policy(policyText)
                    .build();

            s3Client.putBucketPolicy(policyReq);
            logger.info("Bucket policy set successfully!");

        } catch (S3Exception e) {
            logger.error("Failed to set bucket policy: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[PutBucketPolicy](#)中的。

PutObject

以下代码示例演示了如何使用 PutObject。

适用于 Java 的 SDK 2.x

> **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。

将对象放入目录存储桶。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.awscore.exception.AwsErrorDetails;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.UncheckedIOException;
import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;

    /**
     * Puts an object into the specified S3 directory bucket.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be placed in the bucket
     * @param filePath   The path of the file to be uploaded
     */
    public static void putDirectoryBucketObject(S3Client s3Client, String
 bucketName, String objectKey, Path filePath) {
```

```
        logger.info("Putting object: {} into bucket: {}", objectKey, bucketName);

        try {
            // Create a PutObjectRequest
            PutObjectRequest putObj = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Upload the object
            s3Client.putObject(putObj, filePath);
            logger.info("Successfully placed {} into bucket {}", objectKey,
bucketName);

        } catch (UncheckedIOException e) {
            throw S3Exception.builder().message("Failed to read the file: " +
e.getMessage()).cause(e)
                    .awsErrorDetails(AwsErrorDetails.builder()
                            .errorCode("ClientSideException:FailedToReadFile")
                            .errorMessage(e.getMessage())
                            .build())
                    .build();
        } catch (S3Exception e) {
            logger.error("Failed to put object: {}", e.getMessage(), e);
            throw e;
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutObject中的。

`UploadPart`

以下代码示例演示了如何使用 `UploadPart`。

适用于 Java 的 SDK 2.x

> **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。

上传目录存储桶的分段上传的分段。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.nio.ByteBuffer;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.List;

import static
 com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;

    /**
     * This method creates part requests and uploads individual parts to S3.
     * While it uses the UploadPart API to upload a single part, it does so
     * sequentially to handle multiple parts of a file, returning all the completed
     * parts.
     *
     * @param s3Client   The S3 client used to interact with S3
     * @param bucketName The name of the directory bucket
     * @param objectKey  The key (name) of the object to be uploaded
     * @param uploadId   The upload ID used to track the multipart upload
     * @param filePath   The path to the file to be uploaded
     * @return A list of uploaded parts
     * @throws IOException if an I/O error occurs
```

```
     */
    public static List<CompletedPart> multipartUploadForDirectoryBucket(S3Client
s3Client, String bucketName,
            String objectKey, String uploadId, Path filePath) throws IOException {
        logger.info("Uploading parts for object: {} in bucket: {}", objectKey,
bucketName);

        int partNumber = 1;
        List<CompletedPart> uploadedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

        // Read the local file, break down into chunks and process
        try (RandomAccessFile file = new RandomAccessFile(filePath.toFile(), "r")) {
            long fileSize = file.length();
            int position = 0;

            // Sequentially upload parts of the file
            while (position < fileSize) {
                file.seek(position);
                int read = file.getChannel().read(bb);

                bb.flip(); // Swap position and limit before reading from the buffer
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                        .bucket(bucketName)
                        .key(objectKey)
                        .uploadId(uploadId)
                        .partNumber(partNumber)
                        .build();

                UploadPartResponse partResponse = s3Client.uploadPart(
                        uploadPartRequest,
                        RequestBody.fromByteBuffer(bb));

                // Build the uploaded part
                CompletedPart uploadedPart = CompletedPart.builder()
                        .partNumber(partNumber)
                        .eTag(partResponse.eTag())
                        .build();

                // Add the uploaded part to the list
                uploadedParts.add(uploadedPart);

                // Log to indicate the part upload is done
```

```
                logger.info("Uploaded part number: {} with ETag: {}", partNumber,
partResponse.eTag());

                bb.clear();
                position += read;
                partNumber++;
            }
        } catch (S3Exception e) {
            logger.error("Failed to list parts: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
            throw e;
        }
        return uploadedParts;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考UploadPart中的。

`UploadPartCopy`

以下代码示例演示了如何使用 `UploadPartCopy`。

适用于 Java 的 SDK 2.x

> **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。

根据源对象大小创建复制分段，然后将各个分段复制到目录存储桶。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
```

```
import software.amazon.awssdk.services.s3.model.UploadPartCopyRequest;
import software.amazon.awssdk.services.s3.model.UploadPartCopyResponse;

import java.io.IOException;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.List;

import static
 com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static
 com.example.s3.util.S3DirectoryBucketUtils.completeDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
 com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
 com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

    /**
     * Creates copy parts based on source object size and copies over individual
     * parts.
     *
     * @param s3Client          The S3 client used to interact with S3
     * @param sourceBucket      The name of the source bucket
     * @param sourceKey         The key (name) of the source object
     * @param destinationBucket The name of the destination bucket
     * @param destinationKey    The key (name) of the destination object
     * @param uploadId          The upload ID used to track the multipart upload
     * @return A list of completed parts
     */
    public static List<CompletedPart> multipartUploadCopyForDirectoryBucket(S3Client
 s3Client, String sourceBucket,
            String sourceKey, String destinationBucket, String destinationKey,
 String uploadId) {
        // Get the object size to track the end of the copy operation
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(sourceBucket)
                .key(sourceKey)
                .build();
```

```
        HeadObjectResponse headObjectResponse =
s3Client.headObject(headObjectRequest);
        long objectSize = headObjectResponse.contentLength();

        logger.info("Source Object size: {}", objectSize);

        // Copy the object using 20 MB parts
        long partSize = 20 * 1024 * 1024; // 20 MB
        long bytePosition = 0;
        int partNum = 1;
        List<CompletedPart> uploadedParts = new ArrayList<>();

        while (bytePosition < objectSize) {
            long lastByte = Math.min(bytePosition + partSize - 1, objectSize - 1);
            logger.info("Part Number: {}, Byte Position: {}, Last Byte: {}",
partNum, bytePosition, lastByte);

            try {
                UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
                        .sourceBucket(sourceBucket)
                        .sourceKey(sourceKey)
                        .destinationBucket(destinationBucket)
                        .destinationKey(destinationKey)
                        .uploadId(uploadId)
                        .copySourceRange("bytes=" + bytePosition + "-" + lastByte)
                        .partNumber(partNum)
                        .build();
                UploadPartCopyResponse uploadPartCopyResponse =
s3Client.uploadPartCopy(uploadPartCopyRequest);

                CompletedPart part = CompletedPart.builder()
                        .partNumber(partNum)
                        .eTag(uploadPartCopyResponse.copyPartResult().eTag())
                        .build();
                uploadedParts.add(part);

                bytePosition += partSize;
                partNum++;
            } catch (S3Exception e) {
                logger.error("Failed to copy part number {}: {} - Error code: {}",
partNum,
                        e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode());
```

```
                throw e;
            }
        }

        return uploadedParts;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[UploadPartCopy](#)中的。

场景

创建预签名 URL 以获取对象

以下代码示例演示如何为 S3 目录存储桶创建预签名 URL 以及获取对象。

适用于 Java 的 SDK 2.x

> **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。

生成用于访问 S3 目录存储桶中的对象的预签名 GET URL。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;

import java.nio.file.Path;
import java.time.Duration;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Presigner;
import static
 com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;


   /**
    * Generates a presigned URL for accessing an object in the specified S3
    * directory bucket.
    *
    * @param s3Presigner The S3 presigner client used to generate the presigned URL
    * @param bucketName  The name of the directory bucket
    * @param objectKey   The key (name) of the object to access
    * @return A presigned URL for accessing the specified object
    */
   public static String generatePresignedGetURLForDirectoryBucket(S3Presigner
 s3Presigner, String bucketName,
           String objectKey) {
       logger.info("Generating presigned URL for object: {} in bucket: {}",
 objectKey, bucketName);

       try {
           // Create a GetObjectRequest
           GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                   .bucket(bucketName)
                   .key(objectKey)
                   .build();

           // Create a GetObjectPresignRequest
           GetObjectPresignRequest getObjectPresignRequest =
 GetObjectPresignRequest.builder()
                   .signatureDuration(Duration.ofMinutes(10)) // Presigned URL
 valid for 10 minutes
                   .getObjectRequest(getObjectRequest)
                   .build();

           // Generate the presigned URL
           PresignedGetObjectRequest presignedGetObjectRequest =
 s3Presigner.presignGetObject(getObjectPresignRequest);

           // Get the presigned URL
           String presignedURL = presignedGetObjectRequest.url().toString();
```

```
            logger.info("Presigned URL: {}", presignedURL);
            return presignedURL;

        } catch (S3Exception e) {
            logger.error("Failed to generate presigned URL: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetObject中的。

SageMaker 使用适用于 Java 的 SDK 2.x 的人工智能示例

以下代码示例向您展示了如何使用 with SageMaker AI 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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 操作
- 场景

开始使用

你好 SageMaker AI

以下代码示例显示了如何开始使用 SageMaker AI。

适用于 Java 的 SDK 2.x

> **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。

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListNotebookInstances中的。

操作

`CreatePipeline`

以下代码示例演示了如何使用 `CreatePipeline`。

适用于 Java 的 SDK 2.x

> **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。

```
   // 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreatePipeline中的。

`DeletePipeline`

以下代码示例演示了如何使用 `DeletePipeline`。

适用于 Java 的 SDK 2.x

> **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。

```
    // 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeletePipeline中的。

`DescribePipelineExecution`

以下代码示例演示了如何使用 `DescribePipelineExecution`。

适用于 Java 的 SDK 2.x

> **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。

```
   // 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribePipelineExecution中的。

StartPipelineExecution

以下代码示例演示了如何使用 StartPipelineExecution。

适用于 Java 的 SDK 2.x

> **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。

```
    // 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 的详细信息，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。

适用于 Java 的 SDK 2.x

> **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。

```
public class SagemakerWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static String eventSourceMapping = "";

    public static void main(String[] args) throws InterruptedException {
        final String usage = "\n" +
                "Usage:\n" +
                "    <sageMakerRoleName> <lambdaRoleName> <functionFileLocation>
 <functionName> <queueName> <bucketName> <lnglatData> <spatialPipelinePath>
 <pipelineName>\n\n"
                +
                "Where:\n" +
                "    sageMakerRoleName - The name of the Amazon SageMaker role.\n\n"
 +
                "    lambdaRoleName - The name of the AWS Lambda role.\n\n" +
                "    functionFileLocation - The file location where the JAR file
 that represents the AWS Lambda function is located.\n\n"
                +
                "    functionName - The name of the AWS Lambda function (for
 example,SageMakerExampleFunction).\n\n" +
                "    queueName - The name of the Amazon Simple Queue Service (Amazon
 SQS) queue.\n\n" +
                "    bucketName - The name of the Amazon Simple Storage Service
 (Amazon S3) bucket.\n\n" +
                "    lnglatData - The file location of the latlongtest.csv file
 required for this use case.\n\n" +
                "    spatialPipelinePath - The file location of the
 GeoSpatialPipeline.json file required for this use case.\n\n"
                +
                "    pipelineName - The name of the pipeline to create (for example,
 sagemaker-sdk-example-pipeline).\n\n";

        if (args.length != 9) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String sageMakerRoleName = args[0];
        String lambdaRoleName = args[1];
        String functionFileLocation = args[2];
        String functionName = args[3];
        String queueName = args[4];
        String bucketName = args[5];
        String lnglatData = args[6];
        String spatialPipelinePath = args[7];
        String pipelineName = args[8];
        String handlerName = "org.example.SageMakerLambdaFunction::handleRequest";

        Region region = Region.US_WEST_2;
        SageMakerClient sageMakerClient = SageMakerClient.builder()
                .region(region)
                .build();

        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        LambdaClient lambdaClient = LambdaClient.builder()
                .region(region)
                .build();

        SqsClient sqsClient = SqsClient.builder()
                .region(region)
                .build();

        S3Client s3Client = S3Client.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon SageMaker pipeline example
scenario.");
        System.out.println(
                "\nThis example workflow will guide you through setting up and
running an" +
                        "\nAmazon SageMaker pipeline. The pipeline uses an AWS
Lambda function and an" +
```

```
                        "\nAmazon SQS Queue. It runs a vector enrichment reverse
geocode job to" +
                        "\nreverse geocode addresses in an input file and store the
results in an export file.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("First, we will set up the roles, functions, and queue
needed by the SageMaker pipeline.");
        String lambdaRoleArn = checkLambdaRole(iam, lambdaRoleName);
        String sageMakerRoleArn = checkSageMakerRole(iam, sageMakerRoleName);

        String functionArn = checkFunction(lambdaClient, functionName,
functionFileLocation, lambdaRoleArn,
                handlerName);
        String queueUrl = checkQueue(sqsClient, lambdaClient, queueName,
functionName);
        System.out.println("The queue URL is " + queueUrl);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Setting up bucket " + bucketName);
        if (!checkBucket(s3Client, bucketName)) {
            setupBucket(s3Client, bucketName);
            System.out.println("Put " + lnglatData + " into " + bucketName);
            putS3Object(s3Client, bucketName, "latlongtest.csv", lnglatData);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Now we can create and run our pipeline.");
        setupPipeline(sageMakerClient, spatialPipelinePath, sageMakerRoleArn,
functionArn, pipelineName);
        String pipelineExecutionARN = executePipeline(sageMakerClient, bucketName,
queueUrl, sageMakerRoleArn,
                pipelineName);
        System.out.println("The pipeline execution ARN value is " +
pipelineExecutionARN);
        waitForPipelineExecution(sageMakerClient, pipelineExecutionARN);
        System.out.println("Getting output results " + bucketName);
        getOutputResults(s3Client, bucketName);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("The pipeline has completed. To view the pipeline and
 runs " +
                "in SageMaker Studio, follow these instructions:" +
                "\nhttps://docs.aws.amazon.com/sagemaker/latest/dg/pipelines-
studio.html");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Do you want to delete the AWS resources used in this
 Workflow? (y/n)");
        Scanner in = new Scanner(System.in);
        String delResources = in.nextLine();
        if (delResources.compareTo("y") == 0) {
            System.out.println("Lets clean up the AWS resources. Wait 30 seconds");
            TimeUnit.SECONDS.sleep(30);
            deleteEventSourceMapping(lambdaClient);
            deleteSQSQueue(sqsClient, queueName);
            listBucketObjects(s3Client, bucketName);
            deleteBucket(s3Client, bucketName);
            deleteLambdaFunction(lambdaClient, functionName);
            deleteLambdaRole(iam, lambdaRoleName);
            deleteSagemakerRole(iam, sageMakerRoleName);
            deletePipeline(sageMakerClient, pipelineName);
        } else {
            System.out.println("The AWS Resources were not deleted!");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("SageMaker pipeline scenario is complete.");
        System.out.println(DASHES);
    }

    private static void readObject(S3Client s3Client, String bucketName, String key)
 {
        System.out.println("Output file contents: \n");
        GetObjectRequest objectRequest = GetObjectRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build();

        ResponseBytes<GetObjectResponse> objectBytes =
 s3Client.getObjectAsBytes(objectRequest);
        byte[] byteArray = objectBytes.asByteArray();
```

```
        String text = new String(byteArray, StandardCharsets.UTF_8);
        System.out.println("Text output: " + text);
    }

    // Display some results from the output directory.
    public static void getOutputResults(S3Client s3Client, String bucketName) {
        System.out.println("Getting output results {bucketName}.");
        ListObjectsRequest listObjectsRequest = ListObjectsRequest.builder()
                .bucket(bucketName)
                .prefix("outputfiles/")
                .build();

        ListObjectsResponse response = s3Client.listObjects(listObjectsRequest);
        List<S3Object> s3Objects = response.contents();
        for (S3Object object : s3Objects) {
            readObject(s3Client, bucketName, object.key());
        }
    }

    // Check the status of a pipeline execution.
    public static void waitForPipelineExecution(SageMakerClient sageMakerClient,
String executionArn)
            throws InterruptedException {
        String status;
        int index = 0;
        do {
            DescribePipelineExecutionRequest pipelineExecutionRequest =
DescribePipelineExecutionRequest.builder()
                    .pipelineExecutionArn(executionArn)
                    .build();

            DescribePipelineExecutionResponse response = sageMakerClient
                    .describePipelineExecution(pipelineExecutionRequest);
            status = response.pipelineExecutionStatusAsString();
            System.out.println(index + ". The Status of the pipeline is " + status);
            TimeUnit.SECONDS.sleep(4);
            index++;
        } while ("Executing".equals(status));
        System.out.println("Pipeline finished with status " + status);
    }

    // Delete a SageMaker pipeline by name.
    public static void deletePipeline(SageMakerClient sageMakerClient, String
pipelineName) {
```

```
        DeletePipelineRequest pipelineRequest = DeletePipelineRequest.builder()
                .pipelineName(pipelineName)
                .build();

        sageMakerClient.deletePipeline(pipelineRequest);
        System.out.println("*** Successfully deleted " + pipelineName);
    }

    // Create a pipeline from the example pipeline JSON.
    public static void setupPipeline(SageMakerClient sageMakerClient, String
filePath, String roleArn,
            String functionArn, String pipelineName) {
        System.out.println("Setting up the pipeline.");
        JSONParser parser = new JSONParser();

        // Read JSON and get pipeline definition.
        try (FileReader reader = new FileReader(filePath)) {
            Object obj = parser.parse(reader);
            JSONObject jsonObject = (JSONObject) obj;
            JSONArray stepsArray = (JSONArray) jsonObject.get("Steps");
            for (Object stepObj : stepsArray) {
                JSONObject step = (JSONObject) stepObj;
                if (step.containsKey("FunctionArn")) {
                    step.put("FunctionArn", functionArn);
                }
            }
            System.out.println(jsonObject);

            // Create the pipeline.
            CreatePipelineRequest pipelineRequest = CreatePipelineRequest.builder()
                    .pipelineDescription("Java SDK example pipeline")
                    .roleArn(roleArn)
                    .pipelineName(pipelineName)
                    .pipelineDefinition(jsonObject.toString())
                    .build();

            sageMakerClient.createPipeline(pipelineRequest);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        } catch (IOException | ParseException e) {
            throw new RuntimeException(e);
        }
```

```
    }

    // Start a pipeline run with job configurations.
    public static String executePipeline(SageMakerClient sageMakerClient, String
 bucketName, String queueUrl,
            String roleArn, String pipelineName) {
        System.out.println("Starting pipeline execution.");
        String inputBucketLocation = "s3://" + bucketName + "/samplefiles/
latlongtest.csv";
        String output = "s3://" + bucketName + "/outputfiles/";
        Gson gson = new GsonBuilder()
                .setFieldNamingPolicy(FieldNamingPolicy.UPPER_CAMEL_CASE)
                .setPrettyPrinting().create();

        // Set up all parameters required to start the pipeline.
        List<Parameter> parameters = new ArrayList<>();
        Parameter para1 = Parameter.builder()
                .name("parameter_execution_role")
                .value(roleArn)
                .build();

        Parameter para2 = Parameter.builder()
                .name("parameter_queue_url")
                .value(queueUrl)
                .build();

        String inputJSON = "{\n" +
                "  \"DataSourceConfig\": {\n" +
                "    \"S3Data\": {\n" +
                "      \"S3Uri\": \"s3://" + bucketName + "/samplefiles/
latlongtest.csv\"\n" +
                "    },\n" +
                "    \"Type\": \"S3_DATA\"\n" +
                "  },\n" +
                "  \"DocumentType\": \"CSV\"\n" +
                "}";

        System.out.println(inputJSON);

        Parameter para3 = Parameter.builder()
                .name("parameter_vej_input_config")
                .value(inputJSON)
                .build();
```

```
        // Create an ExportVectorEnrichmentJobOutputConfig object.
        VectorEnrichmentJobS3Data jobS3Data = VectorEnrichmentJobS3Data.builder()
                .s3Uri(output)
                .build();

        ExportVectorEnrichmentJobOutputConfig outputConfig =
 ExportVectorEnrichmentJobOutputConfig.builder()
                .s3Data(jobS3Data)
                .build();

        String gson4 = gson.toJson(outputConfig);
        Parameter para4 = Parameter.builder()
                .name("parameter_vej_export_config")
                .value(gson4)
                .build();
        System.out.println("parameter_vej_export_config:" +
 gson.toJson(outputConfig));

        // Create a VectorEnrichmentJobConfig object.
        ReverseGeocodingConfig reverseGeocodingConfig =
 ReverseGeocodingConfig.builder()
                .xAttributeName("Longitude")
                .yAttributeName("Latitude")
                .build();

        VectorEnrichmentJobConfig jobConfig = VectorEnrichmentJobConfig.builder()
                .reverseGeocodingConfig(reverseGeocodingConfig)
                .build();

        String para5JSON = "{\"MapMatchingConfig\":null,\"ReverseGeocodingConfig\":
{\"XAttributeName\":\"Longitude\",\"YAttributeName\":\"Latitude\"}}";
        Parameter para5 = Parameter.builder()
                .name("parameter_step_1_vej_config")
                .value(para5JSON)
                .build();

        System.out.println("parameter_step_1_vej_config:" + gson.toJson(jobConfig));
        parameters.add(para1);
        parameters.add(para2);
        parameters.add(para3);
        parameters.add(para4);
        parameters.add(para5);
```

```
        StartPipelineExecutionRequest pipelineExecutionRequest =
StartPipelineExecutionRequest.builder()
                .pipelineExecutionDescription("Created using Java SDK")
                .pipelineExecutionDisplayName(pipelineName + "-example-execution")
                .pipelineParameters(parameters)
                .pipelineName(pipelineName)
                .build();

        StartPipelineExecutionResponse response =
sageMakerClient.startPipelineExecution(pipelineExecutionRequest);
        return response.pipelineExecutionArn();
    }

    public static void deleteEventSourceMapping(LambdaClient lambdaClient) {
        DeleteEventSourceMappingRequest eventSourceMappingRequest =
DeleteEventSourceMappingRequest.builder()
                .uuid(eventSourceMapping)
                .build();

        lambdaClient.deleteEventSourceMapping(eventSourceMappingRequest);
    }

    public static void deleteSagemakerRole(IamClient iam, String roleName) {
        String[] sageMakerRolePolicies = getSageMakerRolePolicies();
        try {
            for (String policy : sageMakerRolePolicies) {
                // First the policy needs to be detached.
                DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
                        .policyArn(policy)
                        .roleName(roleName)
                        .build();

                iam.detachRolePolicy(rolePolicyRequest);
            }

            // Delete the role.
            DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            iam.deleteRole(roleRequest);
            System.out.println("*** Successfully deleted " + roleName);
```

```
        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteLambdaRole(IamClient iam, String roleName) {
        String[] lambdaRolePolicies = getLambdaRolePolicies();
        try {
            for (String policy : lambdaRolePolicies) {
                // First the policy needs to be detached.
                DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
                        .policyArn(policy)
                        .roleName(roleName)
                        .build();

                iam.detachRolePolicy(rolePolicyRequest);
            }

            // Delete the role.
            DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            iam.deleteRole(roleRequest);
            System.out.println("*** Successfully deleted " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Delete the specific AWS Lambda function.
    public static void deleteLambdaFunction(LambdaClient awsLambda, String
functionName) {
        try {
            DeleteFunctionRequest request = DeleteFunctionRequest.builder()
                    .functionName(functionName)
                    .build();

            awsLambda.deleteFunction(request);
            System.out.println("*** " + functionName + " was deleted");
```

```
        } catch (LambdaException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    // Delete the specific S3 bucket.
    public static void deleteBucket(S3Client s3Client, String bucketName) {
        DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();
        s3Client.deleteBucket(deleteBucketRequest);
        System.out.println("*** " + bucketName + " was deleted.");
    }

    public static void listBucketObjects(S3Client s3, String bucketName) {
        try {
            ListObjectsRequest listObjects = ListObjectsRequest
                    .builder()
                    .bucket(bucketName)
                    .build();

            ListObjectsResponse res = s3.listObjects(listObjects);
            List<S3Object> objects = res.contents();
            for (S3Object myValue : objects) {
                System.out.print("\n The name of the key is " + myValue.key());
                deleteBucketObjects(s3, bucketName, myValue.key());
            }

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteBucketObjects(S3Client s3, String bucketName, String
objectName) {
        ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();
        toDelete.add(ObjectIdentifier.builder()
                .key(objectName)
                .build());
        try {
            DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
```

```
                    .bucket(bucketName)
                    .delete(Delete.builder()
                            .objects(toDelete).build())
                    .build();

            s3.deleteObjects(dor);
            System.out.println("*** " + bucketName + " objects were deleted.");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Delete the specific Amazon SQS queue.
    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                    .queueName(queueName)
                    .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                    .queueUrl(queueUrl)
                    .build();

            sqsClient.deleteQueue(deleteQueueRequest);

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void putS3Object(S3Client s3, String bucketName, String objectKey,
String objectPath) {
        try {
            Map<String, String> metadata = new HashMap<>();
            metadata.put("x-amz-meta-myVal", "test");
            PutObjectRequest putOb = PutObjectRequest.builder()
                    .bucket(bucketName)
                    .key("samplefiles/" + objectKey)
                    .metadata(metadata)
                    .build();
```

```
            s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
            System.out.println("Successfully placed " + objectKey + " into bucket "
+ bucketName);

        } catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void setupBucket(S3Client s3Client, String bucketName) {
        try {
            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Set up the SQS queue to use with the pipeline.
    public static String setupQueue(SqsClient sqsClient, LambdaClient lambdaClient,
String queueName,
            String lambdaName) {
        System.out.println("Setting up queue named " + queueName);
        try {
            Map<QueueAttributeName, String> queueAtt = new HashMap<>();
            queueAtt.put(QueueAttributeName.DELAY_SECONDS, "5");
            queueAtt.put(QueueAttributeName.RECEIVE_MESSAGE_WAIT_TIME_SECONDS, "5");
```

```
            queueAtt.put(QueueAttributeName.VISIBILITY_TIMEOUT, "300");
            CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .attributes(queueAtt)
                    .build();

            sqsClient.createQueue(createQueueRequest);
            System.out.println("\nGet queue url");
            GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
            TimeUnit.SECONDS.sleep(15);

            connectLambda(sqsClient, lambdaClient, getQueueUrlResponse.queueUrl(),
lambdaName);
            System.out.println("Queue ready with Url " +
getQueueUrlResponse.queueUrl());
            return getQueueUrlResponse.queueUrl();

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        return "";
    }

    // Connect the queue to the Lambda function as an event source.
    public static void connectLambda(SqsClient sqsClient, LambdaClient lambdaClient,
String queueUrl,
            String lambdaName) {
        System.out.println("Connecting the Lambda function and queue for the
pipeline.");
        String queueArn = "";

        // Specify the attributes to retrieve.
        List<QueueAttributeName> atts = new ArrayList<>();
        atts.add(QueueAttributeName.QUEUE_ARN);
        GetQueueAttributesRequest attributesRequest =
GetQueueAttributesRequest.builder()
                .queueUrl(queueUrl)
                .attributeNames(atts)
                .build();
```

```
        GetQueueAttributesResponse response =
sqsClient.getQueueAttributes(attributesRequest);
        Map<String, String> queueAtts = response.attributesAsStrings();
        for (Map.Entry<String, String> queueAtt : queueAtts.entrySet()) {
            System.out.println("Key = " + queueAtt.getKey() + ", Value = " +
queueAtt.getValue());
            queueArn = queueAtt.getValue();
        }

        CreateEventSourceMappingRequest eventSourceMappingRequest =
CreateEventSourceMappingRequest.builder()
                .eventSourceArn(queueArn)
                .functionName(lambdaName)
                .build();

        CreateEventSourceMappingResponse response1 =
lambdaClient.createEventSourceMapping(eventSourceMappingRequest);
        eventSourceMapping = response1.uuid();
        System.out.println("The mapping between the event source and Lambda function
was successful");
    }

    // Create an AWS Lambda function.
    public static String createLambdaFunction(LambdaClient awsLambda, String
functionName, String filePath, String role,
            String handler) {
        try {
            LambdaWaiter waiter = awsLambda.waiter();
            InputStream is = new FileInputStream(filePath);
            SdkBytes fileToUpload = SdkBytes.fromInputStream(is);
            FunctionCode code = FunctionCode.builder()
                    .zipFile(fileToUpload)
                    .build();

            CreateFunctionRequest functionRequest = CreateFunctionRequest.builder()
                    .functionName(functionName)
                    .description("SageMaker example function.")
                    .code(code)
                    .handler(handler)
                    .runtime(Runtime.JAVA11)
                    .timeout(200)
                    .memorySize(1024)
                    .role(role)
```

```
                .build();

        // Create a Lambda function using a waiter.
        CreateFunctionResponse functionResponse =
awsLambda.createFunction(functionRequest);
        GetFunctionRequest getFunctionRequest = GetFunctionRequest.builder()
                .functionName(functionName)
                .build();
        WaiterResponse<GetFunctionResponse> waiterResponse =
waiter.waitUntilFunctionExists(getFunctionRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("The function ARN is " +
functionResponse.functionArn());
        return functionResponse.functionArn();

    } catch (LambdaException | FileNotFoundException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    return "";
  }

  public static String createSageMakerRole(IamClient iam, String roleName) {
    String[] sageMakerRolePolicies = getSageMakerRolePolicies();
    System.out.println("Creating a role to use with SageMaker.");
    String assumeRolePolicy = "{" +
            "\"Version\": \"2012-10-17\"," +
            "\"Statement\": [{" +
            "\"Effect\": \"Allow\"," +
            "\"Principal\": {" +
            "\"Service\": [" +
            "\"sagemaker.amazonaws.com\"," +
            "\"sagemaker-geospatial.amazonaws.com\"," +
            "\"lambda.amazonaws.com\"," +
            "\"s3.amazonaws.com\"" +
            "]" +
            "}," +
            "\"Action\": \"sts:AssumeRole\"" +
            "}]" +
            "}";

    try {
        CreateRoleRequest request = CreateRoleRequest.builder()
                .roleName(roleName)
```

```
                    .assumeRolePolicyDocument(assumeRolePolicy)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse roleResult = iam.createRole(request);

            // Attach the policies to the role.
            for (String policy : sageMakerRolePolicies) {
                AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(policy)
                        .build();

                iam.attachRolePolicy(attachRequest);
            }

            // Allow time for the role to be ready.
            TimeUnit.SECONDS.sleep(15);
            System.out.println("Role ready with ARN " + roleResult.role().arn());
            return roleResult.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        return "";
    }

    private static String createLambdaRole(IamClient iam, String roleName) {
        String[] lambdaRolePolicies = getLambdaRolePolicies();
        String assumeRolePolicy = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                "\"Service\": [" +
                "\"sagemaker.amazonaws.com\"," +
                "\"sagemaker-geospatial.amazonaws.com\"," +
                "\"lambda.amazonaws.com\"," +
                "\"s3.amazonaws.com\"" +
                "]" +
```

```
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(roleName)
                    .assumeRolePolicyDocument(assumeRolePolicy)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse roleResult = iam.createRole(request);

            // Attach the policies to the role.
            for (String policy : lambdaRolePolicies) {
                AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(policy)
                        .build();

                iam.attachRolePolicy(attachRequest);
            }

            // Allow time for the role to be ready.
            TimeUnit.SECONDS.sleep(15);
            System.out.println("Role ready with ARN " + roleResult.role().arn());
            return roleResult.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());

        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
        return "";
    }

    public static String checkFunction(LambdaClient lambdaClient, String
functionName, String filePath, String role,
            String handler) {
        System.out.println("Create an AWS Lambda function used in this workflow.");
        String functionArn;
```

```
        try {
            // Does this function already exist.
            GetFunctionRequest functionRequest = GetFunctionRequest.builder()
                    .functionName(functionName)
                    .build();

            GetFunctionResponse response =
lambdaClient.getFunction(functionRequest);
            functionArn = response.configuration().functionArn();

        } catch (LambdaException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            functionArn = createLambdaFunction(lambdaClient, functionName, filePath,
role, handler);
        }
        return functionArn;
    }

    // Check to see if the specific S3 bucket exists. If the S3 bucket exists, this
    // method returns true.
    public static boolean checkBucket(S3Client s3, String bucketName) {
        try {
            HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
                    .bucket(bucketName)
                    .build();

            s3.headBucket(headBucketRequest);
            System.out.println(bucketName + " exists");
            return true;

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        return false;
    }

    // Checks to see if the Amazon SQS queue exists. If not, this method creates a
    // new queue
    // and returns the ARN value.
    public static String checkQueue(SqsClient sqsClient, LambdaClient lambdaClient,
String queueName,
            String lambdaName) {
        System.out.println("Creating a queue for this use case.");
        String queueUrl;
```

```
        try {
            GetQueueUrlRequest request = GetQueueUrlRequest.builder()
                    .queueName(queueName)
                    .build();

            GetQueueUrlResponse response = sqsClient.getQueueUrl(request);
            queueUrl = response.queueUrl();
            System.out.println(queueUrl);

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            queueUrl = setupQueue(sqsClient, lambdaClient, queueName, lambdaName);
        }
        return queueUrl;
    }

    // Checks to see if the Lambda role exists. If not, this method creates it.
    public static String checkLambdaRole(IamClient iam, String roleName) {
        System.out.println("Creating a role to for AWS Lambda to use.");
        String roleArn;
        try {
            GetRoleRequest roleRequest = GetRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            GetRoleResponse response = iam.getRole(roleRequest);
            roleArn = response.role().arn();
            System.out.println(roleArn);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            roleArn = createLambdaRole(iam, roleName);
        }
        return roleArn;
    }

    // Checks to see if the SageMaker role exists. If not, this method creates it.
    public static String checkSageMakerRole(IamClient iam, String roleName) {
        System.out.println("Creating a role to for AWS SageMaker to use.");
        String roleArn;
        try {
            GetRoleRequest roleRequest = GetRoleRequest.builder()
                    .roleName(roleName)
                    .build();
```

```
            GetRoleResponse response = iam.getRole(roleRequest);
            roleArn = response.role().arn();
            System.out.println(roleArn);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            roleArn = createSageMakerRole(iam, roleName);
        }
        return roleArn;
    }

    private static String[] getSageMakerRolePolicies() {
        String[] sageMakerRolePolicies = new String[3];
        sageMakerRolePolicies[0] = "arn:aws:iam::aws:policy/
AmazonSageMakerFullAccess";
        sageMakerRolePolicies[1] = "arn:aws:iam::aws:policy/" +
 "AmazonSageMakerGeospatialFullAccess";
        sageMakerRolePolicies[2] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess";
        return sageMakerRolePolicies;
    }

    private static String[] getLambdaRolePolicies() {
        String[] lambdaRolePolicies = new String[5];
        lambdaRolePolicies[0] = "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess";
        lambdaRolePolicies[1] = "arn:aws:iam::aws:policy/AmazonSQSFullAccess";
        lambdaRolePolicies[2] = "arn:aws:iam::aws:policy/service-role/" +
 "AmazonSageMakerGeospatialFullAccess";
        lambdaRolePolicies[3] = "arn:aws:iam::aws:policy/service-role/"
                + "AmazonSageMakerServiceCatalogProductsLambdaServiceRolePolicy";
        lambdaRolePolicies[4] = "arn:aws:iam::aws:policy/service-role/" +
 "AWSLambdaSQSQueueExecutionRole";
        return lambdaRolePolicies;
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - CreatePipeline
  - DeletePipeline
  - DescribePipelineExecution
  - StartPipelineExecution

- UpdatePipeline

使用 SDK for 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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从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`

以下代码示例演示了如何使用 `GetSecretValue`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetSecretValue中的。

使用 SDK for Java 2.x 的 Amazon SES 示例

以下代码示例向您展示了如何在 Amazon SES 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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从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

`ListIdentities`

以下代码示例演示了如何使用 `ListIdentities`。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ses.SesClient;
import software.amazon.awssdk.services.ses.model.ListIdentitiesResponse;
import software.amazon.awssdk.services.ses.model.SesException;
import java.io.IOException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListIdentities {

    public static void main(String[] args) throws IOException {
        Region region = Region.US_WEST_2;
        SesClient client = SesClient.builder()
                .region(region)
                .build();

        listSESIdentities(client);
    }

    public static void listSESIdentities(SesClient client) {
        try {
```

```
            ListIdentitiesResponse identitiesResponse = client.listIdentities();
            List<String> identities = identitiesResponse.identities();
            for (String identity : identities) {
                System.out.println("The identity is " + identity);
            }

        } catch (SesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ListIdentities](#)中的。

`ListTemplates`

以下代码示例演示了如何使用 `ListTemplates`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ListTemplates](#)中的。

`SendEmail`

以下代码示例演示了如何使用 `SendEmail`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SendEmail中的。

`SendTemplatedEmail`

以下代码示例演示了如何使用 `SendTemplatedEmail`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SendTemplatedEmail中的。

场景

创建 Web 应用程序来跟踪 DynamoDB 数据

以下代码示例演示如何创建一个 Web 应用程序，来跟踪 Amazon DynamoDB 表中的工作项，并使用 Amazon Simple Email Service（Amazon SES）来发送报告。

适用于 Java 的 SDK 2.x

展示如何使用 Amazon DynamoDB API 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
- Amazon SES

创建 Web 应用程序来跟踪 Amazon Redshift 数据

以下代码示例演示如何使用 Amazon Redshift 数据库创建用于跟踪和报告工作项的 Web 应用程序。

适用于 Java 的 SDK 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 数据的 Spring REST API 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
- Amazon SES

创建 Aurora Serverless 工作项跟踪器

以下代码示例演示如何创建 Web 应用程序，来跟踪 Amazon Aurora Serverless 数据库中的工作项，以及使用 Amazon Simple Email Service（Amazon SES）发送报告。

适用于 Java 的 SDK 2.x

展示如何创建 Web 应用程序来跟踪与报告存储与 Amazon RDS 数据库的工作项。

有关如何设置查询 Amazon Aurora Serverless 数据的 Spring REST API 以及如何让 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。

有关如何设置和运行使用 JDBC API 的示例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

- Aurora
- Amazon RDS
- Amazon RDS 数据服务
- Amazon SES

检测图像中的 PPE

以下代码示例展示如何构建采用 Amazon Rekognition 来检测图像中的个人防护设备（PPE）的应用程序。

适用于 Java 的 SDK 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
- Amazon SES

检测图像中的对象

以下代码示例演示如何构建采用 Amazon Rekognition 来按类别检测图像中物体的应用程序。

适用于 Java 的 SDK 2.x

展示如何使用 Amazon Rekognition Java API 创建应用程序，该应用程序采用 Amazon Rekognition 来按类别识别位于 Amazon Simple Storage Service (Amazon S3) 存储桶的图像当中的对象。该应用程序使用 Amazon Simple Email Service (Amazon SES) 向管理员发送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
- Amazon SES

检测视频中的人物和对象

以下代码示例演示如何使用 Amazon Rekognition 检测视频中的人物和物体。

适用于 Java 的 SDK 2.x

展示如何使用 Amazon Rekognition Java API 创建应用程序，以检测位于 Amazon Simple Storage Service (Amazon S3) 存储桶的视频当中的人脸和对象。该应用程序使用 Amazon Simple Email Service (Amazon SES) 向管理员发送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
- Amazon SES
- Amazon SNS
- Amazon SQS

使用 Step Functions 调用 Lambda 函数

以下代码示例说明如何创建按顺序调用 AWS Lambda 函数的 AWS Step Functions 状态机。

适用于 Java 的 SDK 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
- Amazon SES
- Step Functions

使用 SDK for Java 2.x 的 Amazon SES API v2 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 SES API v2 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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从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`

以下代码示例演示了如何使用 `CreateContact`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateContact中的。

`CreateContactList`

以下代码示例演示了如何使用 `CreateContactList`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateContactList中的。

CreateEmailIdentity

以下代码示例演示了如何使用 CreateEmailIdentity。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateEmailIdentity中的。

CreateEmailTemplate

以下代码示例演示了如何使用 CreateEmailTemplate。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateEmailTemplate中的。

`DeleteContactList`

以下代码示例演示了如何使用 `DeleteContactList`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteContactList中的。

DeleteEmailIdentity

以下代码示例演示了如何使用 DeleteEmailIdentity。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteEmailIdentity中的。

DeleteEmailTemplate

以下代码示例演示了如何使用 DeleteEmailTemplate。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteEmailTemplate中的。

`ListContacts`

以下代码示例演示了如何使用 `ListContacts`。

适用于 Java 的 SDK 2.x

> **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。

```
     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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ListContacts](#)中的。

`SendEmail`

以下代码示例演示了如何使用 `SendEmail`。

适用于 Java 的 SDK 2.x

> **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。

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.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class SendEmail {
    public static void main(String[] args) {
        final String usage = """

                             Usage:
                                 <sender> <recipient> <subject>\s

                             Where:
                                 sender - An email address that represents the
 sender.\s
                                 recipient - An email address that represents the
 recipient.\s
                                 subject - The subject line.\s
                             """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];

        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
                .region(region)
                .build();

        // The HTML body of the email.
```

```
        String bodyHTML = "<html>" + "<head></head>" + "<body>" + "<h1>Hello!</h1>"
                + "<p> See the list of customers.</p>" + "</body>" + "</html>";

        send(sesv2Client, sender, recipient, subject, bodyHTML);
    }

    public static void send(SesV2Client client,
                            String sender,
                            String recipient,
                            String subject,
                            String bodyHTML) {

        Destination destination = Destination.builder()
                .toAddresses(recipient)
                .build();

        Content content = Content.builder()
                .data(bodyHTML)
                .build();

        Content sub = Content.builder()
                .data(subject)
                .build();

        Body body = Body.builder()
                .html(content)
                .build();

        Message msg = Message.builder()
                .subject(sub)
                .body(body)
                .build();

        EmailContent emailContent = EmailContent.builder()
                .simple(msg)
                .build();

        SendEmailRequest emailRequest = SendEmailRequest.builder()
                .destination(destination)
                .content(emailContent)
                .fromEmailAddress(sender)
                .build();

        try {
```

```
            System.out.println("Attempting to send an email through Amazon SES "
                    + "using the AWS SDK for Java...");
            client.sendEmail(emailRequest);
            System.out.println("email was sent");

        } catch (SesV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

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());
      }
```

发送带有标头信息的消息。

```
public class SendwithHeader {

    public static void main(String[] args) {
```

```
        final String usage = """

            Usage:
                <sender> <recipient> <subject>\s

            Where:
                sender - An email address that represents the sender.\s
                recipient - An email address that represents the recipient.\s
                subject - The subject line.\s
            """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String sender = args[0];
        String recipient = args[1];
        String subject = args[2];
        Region region = Region.US_EAST_1;
        SesV2Client sesv2Client = SesV2Client.builder()
                .region(region)
                .build();

        String bodyHTML = """
                <html>
                    <head></head>
                    <body>
                        <h1>Hello!</h1>
                        <p>See the list of customers.</p>
                    </body>
                </html>
                """;

        sendWithHeader(sesv2Client, sender, recipient, subject, bodyHTML);
        sesv2Client.close();
    }

    /**
     * Sends an email using the AWS SES V2 client.
     *
     * @param sesv2Client the SES V2 client to use for sending the email
     * @param sender the email address of the sender
     * @param recipient the email address of the recipient
```

```
     * @param subject the subject of the email
     * @param bodyHTML the HTML content of the email body
     */
    public static void sendWithHeader(SesV2Client sesv2Client,
                                      String sender,
                                      String recipient,
                                      String subject,
                                      String bodyHTML) {
        EmailContent emailContent = EmailContent.builder()
                .simple(Message.builder()
                        .body(b -> b.html(c ->
 c.charset(UTF_8.name()).data(bodyHTML))
                                .text(c -> c.charset(UTF_8.name()).data(bodyHTML)))
                        .subject(c -> c.charset(UTF_8.name()).data(subject))
                        .headers(List.of(
                                MessageHeader.builder()
                                        .name("List-Unsubscribe")
                                        .value("<https://nutrition.co/?
address=x&topic=x>, <mailto:unsubscribe@nutrition.co?subject=TopicUnsubscribe>")
                                        .build(),
                                MessageHeader.builder()
                                        .name("List-Unsubscribe-Post")
                                        .value("List-Unsubscribe=One-Click")
                                        .build()))
                        .build())
                .build();

        SendEmailRequest request = SendEmailRequest.builder()
                .fromEmailAddress(sender)
                .destination(d -> d.toAddresses(recipient))
                .content(emailContent)
                .build();

        try {
            SendEmailResponse response = sesv2Client.sendEmail(request);
            System.out.println("Email sent! Message ID: " + response.messageId());
        } catch (SesV2Exception e) {
            System.err.println("Failed to send email: " +
 e.awsErrorDetails().errorMessage());
            throw new RuntimeException(e);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SendEmail中的。

场景

时事通讯场景

以下代码示例演示如何运行 Amazon SES API v2 时事通讯场景。

适用于 Java 的 SDK 2.x

> **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。

```
    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《AWS SDK for Java 2.x API Reference》中的以下主题。
  - CreateContact
  - CreateContactList
  - CreateEmailIdentity
  - CreateEmailTemplate
  - DeleteContactList
  - DeleteEmailIdentity
  - DeleteEmailTemplate

- ListContacts
- SendEmail. 简单
- SendEmail。模板

使用 SDK for Java 2.x 的 Amazon SNS 示例

以下代码示例向您展示了如何在 Amazon SNS 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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 操作
- 场景
- 无服务器示例

开始使用

开始使用 Amazon SNS

以下代码示例显示了如何开始使用 Amazon SNS。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListTopics中的。

操作

`CheckIfPhoneNumberIsOptedOut`

以下代码示例演示了如何使用 `CheckIfPhoneNumberIsOptedOut`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[CheckIfPhoneNumberIsOptedOut](#)中的。

ConfirmSubscription

以下代码示例演示了如何使用 ConfirmSubscription。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ConfirmSubscription中的。

`CreateTopic`

以下代码示例演示了如何使用 `CreateTopic`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateTopic中的。

`DeleteTopic`

以下代码示例演示了如何使用 `DeleteTopic`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteTopic中的。

GetSMSAttributes

以下代码示例演示了如何使用 GetSMSAttributes。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.Iterator;
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetSMSAtrributes {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn>

                Where:
                   topicArn - The ARN of the topic from which to retrieve
 attributes.
```

```
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        getSNSAttrutes(snsClient, topicArn);
        snsClient.close();
    }

    public static void getSNSAttrutes(SnsClient snsClient, String topicArn) {
        try {
            GetSubscriptionAttributesRequest request =
 GetSubscriptionAttributesRequest.builder()
                    .subscriptionArn(topicArn)
                    .build();

            // Get the Subscription attributes
            GetSubscriptionAttributesResponse res =
 snsClient.getSubscriptionAttributes(request);
            Map<String, String> map = res.attributes();

            // Iterate through the map
            Iterator iter = map.entrySet().iterator();
            while (iter.hasNext()) {
                Map.Entry entry = (Map.Entry) iter.next();
                System.out.println("[Key] : " + entry.getKey() + " [Value] : " +
 entry.getValue());
            }

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        System.out.println("\n\nStatus was good");
    }
}
```

- 有关 API 的详细信息，请参阅 Get SMSAttributes in AWS SDK for Java 2.x API 参考。

GetTopicAttributes

以下代码示例演示了如何使用 GetTopicAttributes。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetTopicAttributes中的。

`ListPhoneNumbersOptedOut`

以下代码示例演示了如何使用 `ListPhoneNumbersOptedOut`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ListPhoneNumbersOptedOut](#)中的。

`ListSubscriptions`

以下代码示例演示了如何使用 `ListSubscriptions`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListSubscriptions中的。

`ListTopics`

以下代码示例演示了如何使用 `ListTopics`。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListTopicsRequest;
import software.amazon.awssdk.services.sns.model.ListTopicsResponse;
```

```
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListTopics {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsRequest request = ListTopicsRequest.builder()
                    .build();

            ListTopicsResponse result = snsClient.listTopics(request);
            System.out.println(
                    "Status was " + result.sdkHttpResponse().statusCode() + "\n
\nTopics\n\n" + result.topics());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ListTopics](#)中的。

Publish

以下代码示例演示了如何使用 Publish。

适用于 Java 的 SDK 2.x

> **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。

```
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《AWS SDK for Java 2.x API Reference》中的 Publish。

SetSMSAttributes

以下代码示例演示了如何使用 SetSMSAttributes。

适用于 Java 的 SDK 2.x

> **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。

```
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);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SMSAttributes中的设置。

`SetSubscriptionAttributes`

以下代码示例演示了如何使用 `SetSubscriptionAttributes`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SetSubscriptionAttributes中的。

SetTopicAttributes

以下代码示例演示了如何使用 SetTopicAttributes。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SetTopicAttributes中的。

Subscribe

以下代码示例演示了如何使用 Subscribe。

适用于 Java 的 SDK 2.x

> **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。

将电子邮件地址订阅到主题。

```
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);
        }
    }
}
```

将 HTTP 端点订阅到主题。

```
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);
        }
    }
}
```

将 Lambda 函数订阅到主题。

```
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 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 Reference》中的 Subscribe。

`TagResource`

以下代码示例演示了如何使用 `TagResource`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考TagResource中的。

Unsubscribe

以下代码示例演示了如何使用 Unsubscribe。

适用于 Java 的 SDK 2.x

> **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。

```
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 Reference》中的 Unsubscribe。

场景

构建应用程序以将数据提交到 DynamoDB 表

以下代码示例演示如何构建将数据提交到 Amazon DynamoDB 表并在用户更新该表时通知您的应用程序。

适用于 Java 的 SDK 2.x

展示如何创建动态 Web 应用程序，该应用程序使用 Amazon DynamoDB Java API 提交数据并使用 Amazon Simple Notification Service Java API 发送文本消息。

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。

本示例中使用的服务

- DynamoDB
- Amazon SNS

构建 Amazon SNS 应用程序

以下代码示例演示如何创建具有订阅和发布功能并能转换消息的应用程序。

适用于 Java 的 SDK 2.x

展示如何使用 Amazon Simple Notification Service 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 API 的示例的完整源代码和说明，请参阅上的GitHub完整示例。

本示例中使用的服务

- Amazon SNS
- Amazon Translate

为推送通知创建平台终端节点

以下代码示例演示如何为 Amazon SNS 推送通知创建平台端点。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointRequest;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 *
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
 */

public class RegistrationExample {
    public static void main(String[] args) {
        final String usage = """

            Usage:     <token> <platformApplicationArn>

            Where:
               token - The device token or registration ID of the mobile device.
 This is a unique
               identifier provided by the device platform (e.g., Apple Push
 Notification Service (APNS) for iOS devices, Firebase Cloud Messaging (FCM)
               for Android devices) when the mobile app is registered to receive
 push notifications.

               platformApplicationArn - The ARN value of platform application. You
 can get this value from the AWS Management Console.\s

            """;

        if (args.length != 2) {
            System.out.println(usage);
            return;
        }

        String token = args[0];
        String platformApplicationArn = args[1];
```

```
        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

        createEndpoint(snsClient, token, platformApplicationArn);
    }
    public static void createEndpoint(SnsClient snsClient, String token, String
 platformApplicationArn) {
        System.out.println("Creating platform endpoint with token " + token);
        try {
            CreatePlatformEndpointRequest endpointRequest =
 CreatePlatformEndpointRequest.builder()
                .token(token)
                .platformApplicationArn(platformApplicationArn)
                .build();

            CreatePlatformEndpointResponse response =
 snsClient.createPlatformEndpoint(endpointRequest);
            System.out.println("The ARN of the endpoint is " +
 response.endpointArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }
}
```

创建无服务器应用程序来管理照片

以下代码示例演示如何创建无服务器应用程序，让用户能够使用标签管理照片。

适用于 Java 的 SDK 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 Gateway

- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

创建并发布到 FIFO 主题

以下代码示例显示了如何创建并发布到 Amazon SNS FIFO 主题。

适用于 Java 的 SDK 2.x

> **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。

此示例

- 创建一个 Amazon SNS FIFO 主题、两个 Amazon SQS FIFO 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",
                "FifoThroughputScope", "MessageGroup");

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
                .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            String topicArn = response.topicArn();
            System.out.println("The topic ARN is" + topicArn);

            return topicArn;

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void subscribeQueues(List<QueueData> queues, String topicARN) {
        queues.forEach(queue -> {
            SubscribeRequest subscribeRequest = SubscribeRequest.builder()
                .topicArn(topicARN)
                .endpoint(queue.queueARN)
                .protocol("sqs")
                .build();

            // Subscribe to the endpoint by using the SNS service client.
            // Only Amazon SQS queues can receive notifications from an Amazon SNS
FIFO
            // topic.
            SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
            System.out.println("The queue [" + queue.queueARN + "] subscribed to the
topic [" + topicARN + "]");
            queue.subscriptionARN = subscribeResponse.subscriptionArn();
        });
```

```
    }

    public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

        try {
            // Create and publish a message that updates the wholesale price.
            String subject = "Price Update";
            String dedupId = UUID.randomUUID().toString();
            String attributeName = "business";
            String attributeValue = "wholesale";

            MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
                .dataType("String")
                .stringValue(attributeValue)
                .build();

            Map<String, MessageAttributeValue> attributes = new HashMap<>();
            attributes.put(attributeName, msgAttValue);
            PublishRequest pubRequest = PublishRequest.builder()
                .topicArn(topicArn)
                .subject(subject)
                .message(payload)
                .messageGroupId(groupId)
                .messageDeduplicationId(dedupId)
                .messageAttributes(attributes)
                .build();

            final PublishResponse response = snsClient.publish(pubRequest);
            System.out.println(response.messageId());
            System.out.println(response.sequenceNumber());
            System.out.println("Message was published to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - CreateTopic
  - 发布

- Subscribe

检测视频中的人物和对象

以下代码示例演示如何使用 Amazon Rekognition 检测视频中的人物和物体。

适用于 Java 的 SDK 2.x

展示如何使用 Amazon Rekognition Java API 创建应用程序，以检测位于 Amazon Simple Storage Service (Amazon S3) 存储桶的视频当中的人脸和对象。该应用程序使用 Amazon Simple Email Service (Amazon SES) 向管理员发送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
- Amazon SES
- Amazon SNS
- Amazon SQS

将短信发布到主题

以下代码示例显示了如何：

- 创建 Amazon SNS 主题。
- 使用手机号码订阅主题。
- 向主题发布 SMS 消息，以使所有订阅的电话号码一次接收消息。

适用于 Java 的 SDK 2.x

> **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。

创建一个主题并返回其 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用户。

```
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 SMS 文本消息

以下代码示例演示了如何使用 Amazon SNS 发布 SMS 消息。

适用于 Java 的 SDK 2.x

> **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。

```
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《AWS SDK for Java 2.x API Reference》中的 Publish。

将消息发布到队列

以下代码示例演示了操作流程：

- 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。

适用于 Java 的 SDK 2.x

> **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。

```
package com.example.sns;

import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
```

```
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.MessageAttributeValue;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

import com.google.gson.Gson;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonPrimitive;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
```

```
 * <p>
 * This Java example performs these tasks:
 * <p>
 * 1. Gives the user three options to choose from.
 * 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 * 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 * 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 * 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 * 6. Subscribes to the SQS queue.
 * 7. Publishes a message to the topic.
 * 8. Displays the messages.
 * 9. Deletes the received message.
 * 10. Unsubscribes from the topic.
 * 11. Deletes the SNS topic.
 */
public class SNSWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:\n" +
            "    <fifoQueueARN>\n\n" +
            "Where:\n" +
            "    accountId - Your AWS account Id value.";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

        Scanner in = new Scanner(System.in);
        String accountId = args[0];
        String useFIFO;
        String duplication = "n";
```

```
        String topicName;
        String deduplicationID = null;
        String groupId = null;

        String topicArn;
        String sqsQueueName;
        String sqsQueueUrl;
        String sqsQueueArn;
        String subscriptionArn;
        boolean selectFIFO = false;

        String message;
        List<Message> messageList;
        List<String> filterList = new ArrayList<>();
        String msgAttValue = "";

        System.out.println(DASHES);
        System.out.println("Welcome to messaging with topics and queues.");
        System.out.println("In this scenario, you will create an SNS topic and
 subscribe an SQS queue to the topic.\n" +
            "You can select from several options for configuring the topic and the
 subscriptions for the queue.\n" +
            "You can then post to the topic and see the results in the queue.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("SNS topics can be configured as FIFO (First-In-First-
Out).\n" +
            "FIFO topics deliver messages in order and support deduplication and
 message filtering.\n" +
            "Would you like to work with FIFO topics? (y/n)");
        useFIFO = in.nextLine();
        if (useFIFO.compareTo("y") == 0) {
            selectFIFO = true;
            System.out.println("You have selected FIFO");
            System.out.println(" Because you have chosen a FIFO topic, deduplication
 is supported.\n" +
                "        Deduplication IDs are either set in the message or
 automatically generated from content using a hash function.\n"
                +
                "        If a message is successfully published to an SNS FIFO
 topic, any message published and determined to have the same deduplication ID,\n"
                +
```

```
                "        within the five-minute deduplication interval, is accepted
 but not delivered.\n" +
                "        For more information about deduplication, see https://
docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.");

            System.out.println(
                "Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)");
            duplication = in.nextLine();
            if (duplication.compareTo("y") == 0) {
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            } else {
                System.out.println("Please enter deduplication Id value");
                deduplicationID = in.nextLine();
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            }
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a topic.");
        System.out.println("Enter a name for your SNS topic.");
        topicName = in.nextLine();
        if (selectFIFO) {
            System.out.println("Because you have selected a FIFO topic, '.fifo' must
 be appended to the topic name.");
            topicName = topicName + ".fifo";
            System.out.println("The name of the topic is " + topicName);
            topicArn = createFIFO(snsClient, topicName, duplication);
            System.out.println("The ARN of the FIFO topic is " + topicArn);

        } else {
            System.out.println("The name of the topic is " + topicName);
            topicArn = createSNSTopic(snsClient, topicName);
            System.out.println("The ARN of the non-FIFO topic is " + topicArn);

        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create an SQS queue.");
        System.out.println("Enter a name for your SQS queue.");
```

```
        sqsQueueName = in.nextLine();
        if (selectFIFO) {
            sqsQueueName = sqsQueueName + ".fifo";
        }
        sqsQueueUrl = createQueue(sqsClient, sqsQueueName, selectFIFO);
        System.out.println("The queue URL is " + sqsQueueUrl);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the SQS queue ARN attribute.");
        sqsQueueArn = getSQSQueueAttrs(sqsClient, sqsQueueUrl);
        System.out.println("The ARN of the new queue is " + sqsQueueArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Attach an IAM policy to the queue.");

        // Define the policy to use. Make sure that you change the REGION if you are
        // running this code
        // in a different region.
        String policy = """
        {
             "Statement": [
             {
                 "Effect": "Allow",
                         "Principal": {
                     "Service": "sns.amazonaws.com"
                 },
                 "Action": "sqs:SendMessage",
                         "Resource": "arn:aws:sqs:us-east-1:%s:%s",
                         "Condition": {
                     "ArnEquals": {
                         "aws:SourceArn": "arn:aws:sns:us-east-1:%s:%s"
                     }
                 }
             }
             ]
         }
        """.formatted(accountId, sqsQueueName, accountId, topicName);

        setQueueAttr(sqsClient, sqsQueueUrl, policy);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("6. Subscribe to the SQS queue.");
        if (selectFIFO) {
            System.out.println(
                "If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.\n"
                    +
                    "For information about message filtering, see https://
docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n"
                    +
                    "For this example, you can filter messages by a \"tone\"
 attribute.");
            System.out.println("Would you like to filter messages for " +
 sqsQueueName + "'s subscription to the topic "
                + topicName + "?  (y/n)");
            String filterAns = in.nextLine();
            if (filterAns.compareTo("y") == 0) {
                boolean moreAns = false;
                System.out.println("You can filter messages by one or more of the
 following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                while (!moreAns) {
                    System.out.println("Select a number or choose 0 to end.");
                    String ans = in.nextLine();
                    switch (ans) {
                        case "1":
                            filterList.add("cheerful");
                            break;
                        case "2":
                            filterList.add("funny");
                            break;
                        case "3":
                            filterList.add("serious");
                            break;
                        case "4":
                            filterList.add("sincere");
                            break;
                        default:
                            moreAns = true;
                            break;
                    }
                }
```

```
            }
        }
        subscriptionArn = subQueue(snsClient, topicArn, sqsQueueArn, filterList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Publish a message to the topic.");
        if (selectFIFO) {
            System.out.println("Would you like to add an attribute to this message?
(y/n)");
            String msgAns = in.nextLine();
            if (msgAns.compareTo("y") == 0) {
                System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                System.out.println("Select a number or choose 0 to end.");
                String ans = in.nextLine();
                switch (ans) {
                    case "1":
                        msgAttValue = "cheerful";
                        break;
                    case "2":
                        msgAttValue = "funny";
                        break;
                    case "3":
                        msgAttValue = "serious";
                        break;
                    default:
                        msgAttValue = "sincere";
                        break;
                }

                System.out.println("Selected value is " + msgAttValue);
            }
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessageFIFO(snsClient, message, topicArn, msgAttValue, duplication,
groupId, deduplicationID);

        } else {
            System.out.println("Enter a message.");
```

```
            message = in.nextLine();
            pubMessage(snsClient, message, topicArn);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Display the message. Press any key to continue.");
        in.nextLine();
        messageList = receiveMessages(sqsClient, sqsQueueUrl, msgAttValue);
        for (Message mes : messageList) {
            System.out.println("Message Id: " + mes.messageId());
            System.out.println("Full Message: " + mes.body());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Delete the received message. Press any key to
continue.");
        in.nextLine();
        deleteMessages(sqsClient, sqsQueueUrl, messageList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Unsubscribe from the topic and delete the queue.
Press any key to continue.");
        in.nextLine();
        unSub(snsClient, subscriptionArn);
        deleteSQSQueue(sqsClient, sqsQueueName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete the topic. Press any key to continue.");
        in.nextLine();
        deleteSNSTopic(snsClient, topicArn);

        System.out.println(DASHES);
        System.out.println("The SNS/SQS workflow has completed successfully.");
        System.out.println(DASHES);
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                .topicArn(topicArn)
```

```
                .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("Status was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                .queueName(queueName)
                .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                .queueUrl(queueUrl)
                .build();

            sqsClient.deleteQueue(deleteQueueRequest);
            System.out.println(queueName + " was successfully deleted.");

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                .subscriptionArn(subscriptionArn)
                .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("Status was " + result.sdkHttpResponse().statusCode()
                + "\nSubscription was removed for " + request.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }

    public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
        try {
            List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();
            for (Message msg : messages) {
                DeleteMessageBatchRequestEntry entry =
DeleteMessageBatchRequestEntry.builder()
                    .id(msg.messageId())
                    .build();

                entries.add(entry);
            }

            DeleteMessageBatchRequest deleteMessageBatchRequest =
DeleteMessageBatchRequest.builder()
                .queueUrl(queueUrl)
                .entries(entries)
                .build();

            sqsClient.deleteMessageBatch(deleteMessageBatchRequest);
            System.out.println("The batch delete of messages was successful");

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl, String msgAttValue) {
        try {
            if (msgAttValue.isEmpty()) {
                ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .maxNumberOfMessages(5)
                    .build();
                return sqsClient.receiveMessage(receiveMessageRequest).messages();
            } else {
                // We know there are filters on the message.
```

```
                ReceiveMessageRequest receiveRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageAttributeNames(msgAttValue) // Include other message
attributes if needed.
                    .maxNumberOfMessages(5)
                    .build();

                return sqsClient.receiveMessage(receiveRequest).messages();
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static void pubMessage(SnsClient snsClient, String message, String
topicArn) {
        try {
            PublishRequest request = PublishRequest.builder()
                .message(message)
                .topicArn(topicArn)
                .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                .println(result.messageId() + " Message sent. Status is " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void pubMessageFIFO(SnsClient snsClient,
                                      String message,
                                      String topicArn,
                                      String msgAttValue,
                                      String duplication,
                                      String groupId,
                                      String deduplicationID) {
```

```
        try {
            PublishRequest request;
            // Means the user did not choose to use a message attribute.
            if (msgAttValue.isEmpty()) {
                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                } else {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageDeduplicationId(deduplicationID)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                }

            } else {
                Map<String, MessageAttributeValue> messageAttributes = new
HashMap<>();
                messageAttributes.put(msgAttValue, MessageAttributeValue.builder()
                    .dataType("String")
                    .stringValue("true")
                    .build());

                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                } else {
                    // Create a publish request with the message and attributes.
                    request = PublishRequest.builder()
                        .topicArn(topicArn)
                        .message(message)
                        .messageDeduplicationId(deduplicationID)
                        .messageGroupId(groupId)
                        .messageAttributes(messageAttributes)
                        .build();
                }
```

```
            }

            // Publish the message to the topic.
            PublishResponse result = snsClient.publish(request);
            System.out
                .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Subscribe to the SQS queue.
    public static String subQueue(SnsClient snsClient, String topicArn, String
queueArn, List<String> filterList) {
        try {
            SubscribeRequest request;
            if (filterList.isEmpty()) {
                // No filter subscription is added.
                request = SubscribeRequest.builder()
                    .protocol("sqs")
                    .endpoint(queueArn)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been subscribed
to the topic " + topicArn + "\n" +
                    "with the subscription ARN " + result.subscriptionArn());
                return result.subscriptionArn();
            } else {
                request = SubscribeRequest.builder()
                    .protocol("sqs")
                    .endpoint(queueArn)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been subscribed
to the topic " + topicArn + "\n" +
```

```
                    "with the subscription ARN " + result.subscriptionArn());

                String attributeName = "FilterPolicy";
                Gson gson = new Gson();
                String jsonString = "{\"tone\": []}";
                JsonObject jsonObject = gson.fromJson(jsonString, JsonObject.class);
                JsonArray toneArray = jsonObject.getAsJsonArray("tone");
                for (String value : filterList) {
                    toneArray.add(new JsonPrimitive(value));
                }

                String updatedJsonString = gson.toJson(jsonObject);
                System.out.println(updatedJsonString);
                SetSubscriptionAttributesRequest attRequest =
SetSubscriptionAttributesRequest.builder()
                    .subscriptionArn(result.subscriptionArn())
                    .attributeName(attributeName)
                    .attributeValue(updatedJsonString)
                    .build();

                snsClient.setSubscriptionAttributes(attRequest);
                return result.subscriptionArn();
            }

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    // Attach a policy to the queue.
    public static void setQueueAttr(SqsClient sqsClient, String queueUrl, String
policy) {
        try {
            Map<software.amazon.awssdk.services.sqs.model.QueueAttributeName,
String> attrMap = new HashMap<>();
            attrMap.put(QueueAttributeName.POLICY, policy);

            SetQueueAttributesRequest attributesRequest =
SetQueueAttributesRequest.builder()
                .queueUrl(queueUrl)
                .attributes(attrMap)
                .build();
```

```
            sqsClient.setQueueAttributes(attributesRequest);
            System.out.println("The policy has been successfully attached.");

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getSQSQueueAttrs(SqsClient sqsClient, String queueUrl) {
        // Specify the attributes to retrieve.
        List<QueueAttributeName> atts = new ArrayList<>();
        atts.add(QueueAttributeName.QUEUE_ARN);

        GetQueueAttributesRequest attributesRequest =
GetQueueAttributesRequest.builder()
            .queueUrl(queueUrl)
            .attributeNames(atts)
            .build();

        GetQueueAttributesResponse response =
sqsClient.getQueueAttributes(attributesRequest);
        Map<String, String> queueAtts = response.attributesAsStrings();
        for (Map.Entry<String, String> queueAtt : queueAtts.entrySet())
            return queueAtt.getValue();

        return "";
    }

    public static String createQueue(SqsClient sqsClient, String queueName, Boolean
selectFIFO) {
        try {
            System.out.println("\nCreate Queue");
            if (selectFIFO) {
                Map<QueueAttributeName, String> attrs = new HashMap<>();
                attrs.put(QueueAttributeName.FIFO_QUEUE, "true");
                CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .attributes(attrs)
                    .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
```

```
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            } else {
                CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                .name(topicName)
                .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createFIFO(SnsClient snsClient, String topicName, String
duplication) {
        try {
```

```
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = new HashMap<>();
            if (duplication.compareTo("n") == 0) {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "false");
            } else {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "true");
            }

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
                .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            return response.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - 发布
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

使用 API Gateway 调用 Lambda 函数

以下代码示例展示了如何创建由 Amazon API Gateway 调用的 AWS Lambda 函数。

适用于 Java 的 SDK 2.x

演示如何使用 Lambda Java 运行时 API 创建 AWS Lambda 函数。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通过 Amazon API Gateway 调用的 Lambda 函数，该函数扫描 Amazon DynamoDB 表获取工作周年纪念日，并使用 Amazon Simple Notification Service (Amazon SNS)向员工发送文本消息，祝贺他们的周年纪念日。

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 Gateway
- DynamoDB
- Lambda
- Amazon SNS

使用计划的事件调用 Lambda 函数

以下代码示例说明如何创建由 Amazon EventBridge 计划事件调用的 AWS Lambda 函数。

适用于 Java 的 SDK 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。在本示例中，您使用 Lambda Java 运行时 API 创建 Lambda 函数。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

- CloudWatch 日志
- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

无服务器示例

通过 Amazon SNS 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 SNS 主题的消息而触发的事件。该函数从事件参数检索消息并记录每条消息的内容。

适用于 Java 的 SDK 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。

通过 Java 将 SNS 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.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 for Java 2.x 的 Amazon SQS 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 SQS 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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 操作
- 场景
- 无服务器示例

开始使用

开始使用 Amazon SQS

以下代码示例显示了如何开始使用 Amazon SQS。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListQueues中的。

操作

`CreateQueue`

以下代码示例演示了如何使用 `CreateQueue`。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.ChangeMessageVisibilityRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.ListQueuesRequest;
import software.amazon.awssdk.services.sqs.model.ListQueuesResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
```

```
import software.amazon.awssdk.services.sqs.model.SqsException;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SQSExample {
    public static void main(String[] args) {
        String queueName = "queue" + System.currentTimeMillis();
        SqsClient sqsClient = SqsClient.builder()
                .region(Region.US_WEST_2)
                .build();

        // Perform various tasks on the Amazon SQS queue.
        String queueUrl = createQueue(sqsClient, queueName);
        listQueues(sqsClient);
        listQueuesFilter(sqsClient, queueUrl);
        List<Message> messages = receiveMessages(sqsClient, queueUrl);
        sendBatchMessages(sqsClient, queueUrl);
        changeMessages(sqsClient, queueUrl, messages);
        deleteMessages(sqsClient, queueUrl, messages);
        sqsClient.close();
    }

    public static String createQueue(SqsClient sqsClient, String queueName) {
        try {
            System.out.println("\nCreate Queue");

            CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .build();

            sqsClient.createQueue(createQueueRequest);

            System.out.println("\nGet queue url");

            GetQueueUrlResponse getQueueUrlResponse = sqsClient

 .getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
```

```
            return getQueueUrlResponse.queueUrl();

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void listQueues(SqsClient sqsClient) {

        System.out.println("\nList Queues");
        String prefix = "que";

        try {
            ListQueuesRequest listQueuesRequest =
ListQueuesRequest.builder().queueNamePrefix(prefix).build();
            ListQueuesResponse listQueuesResponse =
sqsClient.listQueues(listQueuesRequest);
            for (String url : listQueuesResponse.queueUrls()) {
                System.out.println(url);
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void listQueuesFilter(SqsClient sqsClient, String queueUrl) {
        // List queues with filters
        String namePrefix = "queue";
        ListQueuesRequest filterListRequest = ListQueuesRequest.builder()
                .queueNamePrefix(namePrefix)
                .build();

        ListQueuesResponse listQueuesFilteredResponse =
sqsClient.listQueues(filterListRequest);
        System.out.println("Queue URLs with prefix: " + namePrefix);
        for (String url : listQueuesFilteredResponse.queueUrls()) {
            System.out.println(url);
        }

        System.out.println("\nSend message");
```

```
        try {
            sqsClient.sendMessage(SendMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageBody("Hello world!")
                    .delaySeconds(10)
                    .build());

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void sendBatchMessages(SqsClient sqsClient, String queueUrl) {

        System.out.println("\nSend multiple messages");
        try {
            SendMessageBatchRequest sendMessageBatchRequest =
SendMessageBatchRequest.builder()
                    .queueUrl(queueUrl)

.entries(SendMessageBatchRequestEntry.builder().id("id1").messageBody("Hello from
msg 1").build(),

SendMessageBatchRequestEntry.builder().id("id2").messageBody("msg
2").delaySeconds(10)
                                    .build())
                    .build();
            sqsClient.sendMessageBatch(sendMessageBatchRequest);

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl) {

        System.out.println("\nReceive messages");
        try {
            ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
```

```
                    .maxNumberOfMessages(5)
                    .build();
            return sqsClient.receiveMessage(receiveMessageRequest).messages();

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static void changeMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {

        System.out.println("\nChange Message Visibility");
        try {

            for (Message message : messages) {
                ChangeMessageVisibilityRequest req =
ChangeMessageVisibilityRequest.builder()
                        .queueUrl(queueUrl)
                        .receiptHandle(message.receiptHandle())
                        .visibilityTimeout(100)
                        .build();
                sqsClient.changeMessageVisibility(req);
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
        System.out.println("\nDelete Messages");

        try {
            for (Message message : messages) {
                DeleteMessageRequest deleteMessageRequest =
DeleteMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .receiptHandle(message.receiptHandle())
                        .build();
```

```
                sqsClient.deleteMessage(deleteMessageRequest);
            }
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateQueue中的。

`DeleteMessage`

以下代码示例演示了如何使用 `DeleteMessage`。

适用于 Java 的 SDK 2.x

> **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。

```
        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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteMessage中的。

DeleteQueue

以下代码示例演示了如何使用 DeleteQueue。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteQueue中的。

`GetQueueUrl`

以下代码示例演示了如何使用 `GetQueueUrl`。

适用于 Java 的 SDK 2.x

> **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。

```
            GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
            return getQueueUrlResponse.queueUrl();
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetQueueUrl中的。

`ListQueues`

以下代码示例演示了如何使用 `ListQueues`。

适用于 Java 的 SDK 2.x

> **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。

```
        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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListQueues中的。

ReceiveMessage

以下代码示例演示了如何使用 ReceiveMessage。

适用于 Java 的 SDK 2.x

> **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。

```
        try {
            ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .maxNumberOfMessages(5)
                    .build();
            return sqsClient.receiveMessage(receiveMessageRequest).messages();

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ReceiveMessage中的。

SendMessage

以下代码示例演示了如何使用 SendMessage。

适用于 Java 的 SDK 2.x

> **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。

以下是 SendMessage 操作的两个示例：

- 发送带有正文和延迟的消息
- 发送带有正文和消息属性的消息

发送带有正文和延迟的消息。

```
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);
        }
    }
}
```

发送带有正文和消息属性的消息。

```
    /**
     * <p>This method demonstrates how to add message attributes to a message.
     * Each attribute must specify a name, value, and data type. You use a Java Map
 to supply the attributes. The map's
     * key is the attribute name, and you specify the map's entry value using a
 builder that includes the attribute
     * value and data type.</p>
     *
```

```
     * <p>The data type must start with one of "String", "Number" or "Binary". You
 can optionally
     * define a custom extension by using a "." and your extension.</p>
     *
     * <p>The SQS Developer Guide provides more information on @see <a
     * href="https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-message-metadata.html#sqs-message-attributes">message
     * attributes</a>.</p>
     *
     * @param thumbailPath Filesystem path of the image.
     * @param queueUrl     URL of the SQS queue.
     */
    static void sendMessageWithAttributes(Path thumbailPath, String queueUrl) {
        Map<String, MessageAttributeValue> messageAttributeMap;
        try {
            messageAttributeMap = Map.of(
                    "Name", MessageAttributeValue.builder()
                            .stringValue("Jane Doe")
                            .dataType("String").build(),
                    "Age", MessageAttributeValue.builder()
                            .stringValue("42")
                            .dataType("Number.int").build(),
                    "Image", MessageAttributeValue.builder()

 .binaryValue(SdkBytes.fromByteArray(Files.readAllBytes(thumbailPath)))
                            .dataType("Binary.jpg").build()
            );
        } catch (IOException e) {
            LOGGER.error("An I/O exception occurred reading thumbnail image: {}",
 e.getMessage(), e);
            throw new RuntimeException(e);
        }

        SendMessageRequest request = SendMessageRequest.builder()
                .queueUrl(queueUrl)
                .messageBody("Hello SQS")
                .messageAttributes(messageAttributeMap)
                .build();
        try {
            SendMessageResponse sendMessageResponse =
 SQS_CLIENT.sendMessage(request);
            LOGGER.info("Message ID: {}", sendMessageResponse.messageId());
        } catch (SqsException e) {
```

```
            LOGGER.error("Exception occurred sending message: {}", e.getMessage(),
e);
            throw new RuntimeException(e);
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SendMessage中的。

SendMessageBatch

以下代码示例演示了如何使用 SendMessageBatch。

适用于 Java 的 SDK 2.x

> **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。

```
            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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SendMessageBatch中的。

SetQueueAttributes

以下代码示例演示了如何使用 SetQueueAttributes。

适用于 Java 的 SDK 2.x

> **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。

将 Amazon SQS 配置为使用采用自定义 KMS 密钥的服务器端加密（SSE）。

```
    public static void addEncryption(String queueName, String kmsMasterKeyAlias) {
        SqsClient sqsClient = SqsClient.create();

        GetQueueUrlRequest urlRequest = GetQueueUrlRequest.builder()
                .queueName(queueName)
                .build();

        GetQueueUrlResponse getQueueUrlResponse;
        try {
            getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest);
        } catch (QueueDoesNotExistException e) {
            LOGGER.error(e.getMessage(), e);
            throw new RuntimeException(e);
        }
        String queueUrl = getQueueUrlResponse.queueUrl();


        Map<QueueAttributeName, String> attributes = Map.of(
                QueueAttributeName.KMS_MASTER_KEY_ID, kmsMasterKeyAlias,
                QueueAttributeName.KMS_DATA_KEY_REUSE_PERIOD_SECONDS, "140" // Set
the data key reuse period to 140 seconds.
        );                                                                  // This
is how long SQS can reuse the data key before requesting a new one from KMS.

        SetQueueAttributesRequest attRequest = SetQueueAttributesRequest.builder()
                .queueUrl(queueUrl)
                .attributes(attributes)
                .build();
        try {
            sqsClient.setQueueAttributes(attRequest);
            LOGGER.info("The attributes have been applied to {}", queueName);
        } catch (InvalidAttributeNameException | InvalidAttributeValueException e) {
            LOGGER.error(e.getMessage(), e);
```

```
            throw new RuntimeException(e);
        } finally {
            sqsClient.close();
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SetQueueAttributes中的。

场景

创建消息收发应用程序

以下代码示例演示如何使用 Amazon SQS 创建消息收发应用程序。

适用于 Java 的 SDK 2.x

演示如何使用 Amazon SQS API 开发用于发送和检索消息的 Spring REST 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

- Amazon Comprehend
- Amazon SQS

创建并发布到 FIFO 主题

以下代码示例显示了如何创建并发布到 Amazon SNS FIFO 主题。

适用于 Java 的 SDK 2.x

> **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。

此示例

- 创建一个 Amazon SNS FIFO 主题、两个 Amazon SQS FIFO 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",
                "FifoThroughputScope", "MessageGroup");

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                .name(topicName)
                .attributes(topicAttributes)
                .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            String topicArn = response.topicArn();
            System.out.println("The topic ARN is" + topicArn);

            return topicArn;

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

```
   public static void subscribeQueues(List<QueueData> queues, String topicARN) {
       queues.forEach(queue -> {
           SubscribeRequest subscribeRequest = SubscribeRequest.builder()
               .topicArn(topicARN)
               .endpoint(queue.queueARN)
               .protocol("sqs")
               .build();

           // Subscribe to the endpoint by using the SNS service client.
           // Only Amazon SQS queues can receive notifications from an Amazon SNS
FIFO
           // topic.
           SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
           System.out.println("The queue [" + queue.queueARN + "] subscribed to the
topic [" + topicARN + "]");
           queue.subscriptionARN = subscribeResponse.subscriptionArn();
       });
   }

   public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

       try {
           // Create and publish a message that updates the wholesale price.
           String subject = "Price Update";
           String dedupId = UUID.randomUUID().toString();
           String attributeName = "business";
           String attributeValue = "wholesale";

           MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
               .dataType("String")
               .stringValue(attributeValue)
               .build();

           Map<String, MessageAttributeValue> attributes = new HashMap<>();
           attributes.put(attributeName, msgAttValue);
           PublishRequest pubRequest = PublishRequest.builder()
               .topicArn(topicArn)
               .subject(subject)
               .message(payload)
               .messageGroupId(groupId)
               .messageDeduplicationId(dedupId)
               .messageAttributes(attributes)
```

```
                .build();

            final PublishResponse response = snsClient.publish(pubRequest);
            System.out.println(response.messageId());
            System.out.println(response.sequenceNumber());
            System.out.println("Message was published to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - CreateTopic
  - 发布
  - Subscribe

检测视频中的人物和对象

以下代码示例演示如何使用 Amazon Rekognition 检测视频中的人物和物体。

适用于 Java 的 SDK 2.x

展示如何使用 Amazon Rekognition Java API 创建应用程序，以检测位于 Amazon Simple Storage Service (Amazon S3) 存储桶的视频当中的人脸和对象。该应用程序使用 Amazon Simple Email Service (Amazon SES) 向管理员发送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
- Amazon SES
- Amazon SNS
- Amazon SQS

使用 S3 管理大型消息

以下代码示例演示如何使用 Amazon SQS 扩展型客户端库来处理大型 Amazon SQS 消息。

适用于 Java 的 SDK 2.x

> **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。

```
import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.joda.time.DateTime;
import org.joda.time.format.DateTimeFormat;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.LifecycleExpiration;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueResponse;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
```

```
import java.util.Arrays;
import java.util.List;
import java.util.UUID;

/**
 * Example of using Amazon SQS Extended Client Library for Java 2.x.
 */
public class SqsExtendedClientExample {
    private static final Logger logger =
 LoggerFactory.getLogger(SqsExtendedClientExample.class);

    private String s3BucketName;
    private String queueUrl;
    private final String queueName;
    private final S3Client s3Client;
    private final SqsClient sqsExtendedClient;
    private final int messageSize;

    /**
     * Constructor with default clients and message size.
     */
    public SqsExtendedClientExample() {
        this(S3Client.create(), 300000);
    }

    /**
     * Constructor with custom S3 client and message size.
     *
     * @param s3Client The S3 client to use
     * @param messageSize The size of the test message to create
     */
    public SqsExtendedClientExample(S3Client s3Client, int messageSize) {
        this.s3Client = s3Client;
        this.messageSize = messageSize;

        // Generate a unique bucket name.
        this.s3BucketName = UUID.randomUUID() + "-" +
                DateTimeFormat.forPattern("yyMMdd-hhmmss").print(new DateTime());

        // Generate a unique queue name.
        this.queueName = "MyQueue-" + UUID.randomUUID();

        // Configure the SQS extended client.
```

```
        final ExtendedClientConfiguration extendedClientConfig = new
ExtendedClientConfiguration()
                .withPayloadSupportEnabled(s3Client, s3BucketName);

        this.sqsExtendedClient = new
AmazonSQSExtendedClient(SqsClient.builder().build(), extendedClientConfig);
    }

    public static void main(String[] args) {
        SqsExtendedClientExample example = new SqsExtendedClientExample();
        try {
            example.setup();
            example.sendAndReceiveMessage();
        } finally {
            example.cleanup();
        }
    }

    /**
     * Send a large message and receive it back.
     *
     * @return The received message
     */
    public Message sendAndReceiveMessage() {
        try {
            // Create a large message.
            char[] chars = new char[messageSize];
            Arrays.fill(chars, 'x');
            String largeMessage = new String(chars);

            // Send the message.
            final SendMessageRequest sendMessageRequest =
SendMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageBody(largeMessage)
                    .build();

            sqsExtendedClient.sendMessage(sendMessageRequest);
            logger.info("Sent message of size: {}", largeMessage.length());

            // Receive and return the message.
            final ReceiveMessageResponse receiveMessageResponse =
sqsExtendedClient.receiveMessage(
                    ReceiveMessageRequest.builder().queueUrl(queueUrl).build());
```

```
            List<Message> messages = receiveMessageResponse.messages();
            if (messages.isEmpty()) {
                throw new RuntimeException("No messages received");
            }

            Message message = messages.getFirst();
            logger.info("\nMessage received.");
            logger.info("  ID: {}", message.messageId());
            logger.info("  Receipt handle: {}", message.receiptHandle());
            logger.info("  Message body size: {}", message.body().length());
            logger.info("  Message body (first 5 characters): {}",
message.body().substring(0, 5));

            return message;
        } catch (RuntimeException e) {
            logger.error("Error during message processing: {}", e.getMessage(), e);
            throw e;
        }
    }
```

- 有关更多信息，请参阅《AWS SDK for Java 2.x 开发人员指南》https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-s3-messages.html。
- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - CreateBucket
  - PutBucketLifecycleConfiguration
  - ReceiveMessage
  - SendMessage

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。

适用于 Java 的 SDK 2.x

> **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。

此示例显示了如何使用 Amazon SQS 处理 S3 通知事件。

```
    /**
     * 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《AWS SDK for Java 2.x API Reference》中的以下主题。
  - DeleteMessageBatch
  - GetQueueAttributes

- PutBucketNotificationConfiguration
- ReceiveMessage

将消息发布到队列

以下代码示例演示了操作流程：

- 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。

适用于 Java 的 SDK 2.x

> **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。

```
package com.example.sns;

import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.MessageAttributeValue;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
```

```
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

import com.google.gson.Gson;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonPrimitive;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 * <p>
 * This Java example performs these tasks:
 * <p>
 * 1. Gives the user three options to choose from.
 * 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 * 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 * 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 * 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 * 6. Subscribes to the SQS queue.
 * 7. Publishes a message to the topic.
 * 8. Displays the messages.
```

```
 * 9. Deletes the received message.
 * 10. Unsubscribes from the topic.
 * 11. Deletes the SNS topic.
 */
public class SNSWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");

    public static void main(String[] args) {
        final String usage = "\n" +
            "Usage:\n" +
            "    <fifoQueueARN>\n\n" +
            "Where:\n" +
            "    accountId - Your AWS account Id value.";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

        SqsClient sqsClient = SqsClient.builder()
            .region(Region.US_EAST_1)
            .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
            .build();

        Scanner in = new Scanner(System.in);
        String accountId = args[0];
        String useFIFO;
        String duplication = "n";
        String topicName;
        String deduplicationID = null;
        String groupId = null;

        String topicArn;
        String sqsQueueName;
        String sqsQueueUrl;
        String sqsQueueArn;
        String subscriptionArn;
        boolean selectFIFO = false;
```

```
        String message;
        List<Message> messageList;
        List<String> filterList = new ArrayList<>();
        String msgAttValue = "";

        System.out.println(DASHES);
        System.out.println("Welcome to messaging with topics and queues.");
        System.out.println("In this scenario, you will create an SNS topic and
 subscribe an SQS queue to the topic.\n" +
            "You can select from several options for configuring the topic and the
 subscriptions for the queue.\n" +
            "You can then post to the topic and see the results in the queue.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("SNS topics can be configured as FIFO (First-In-First-
Out).\n" +
            "FIFO topics deliver messages in order and support deduplication and
 message filtering.\n" +
            "Would you like to work with FIFO topics? (y/n)");
        useFIFO = in.nextLine();
        if (useFIFO.compareTo("y") == 0) {
            selectFIFO = true;
            System.out.println("You have selected FIFO");
            System.out.println(" Because you have chosen a FIFO topic, deduplication
 is supported.\n" +
                "        Deduplication IDs are either set in the message or
 automatically generated from content using a hash function.\n"
                +
                "        If a message is successfully published to an SNS FIFO
 topic, any message published and determined to have the same deduplication ID,\n"
                +
                "        within the five-minute deduplication interval, is accepted
 but not delivered.\n" +
                "        For more information about deduplication, see https://
docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.");

            System.out.println(
                "Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)");
            duplication = in.nextLine();
            if (duplication.compareTo("y") == 0) {
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
```

```
            } else {
                System.out.println("Please enter deduplication Id value");
                deduplicationID = in.nextLine();
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            }
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a topic.");
        System.out.println("Enter a name for your SNS topic.");
        topicName = in.nextLine();
        if (selectFIFO) {
            System.out.println("Because you have selected a FIFO topic, '.fifo' must
be appended to the topic name.");
            topicName = topicName + ".fifo";
            System.out.println("The name of the topic is " + topicName);
            topicArn = createFIFO(snsClient, topicName, duplication);
            System.out.println("The ARN of the FIFO topic is " + topicArn);

        } else {
            System.out.println("The name of the topic is " + topicName);
            topicArn = createSNSTopic(snsClient, topicName);
            System.out.println("The ARN of the non-FIFO topic is " + topicArn);

        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create an SQS queue.");
        System.out.println("Enter a name for your SQS queue.");
        sqsQueueName = in.nextLine();
        if (selectFIFO) {
            sqsQueueName = sqsQueueName + ".fifo";
        }
        sqsQueueUrl = createQueue(sqsClient, sqsQueueName, selectFIFO);
        System.out.println("The queue URL is " + sqsQueueUrl);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the SQS queue ARN attribute.");
        sqsQueueArn = getSQSQueueAttrs(sqsClient, sqsQueueUrl);
        System.out.println("The ARN of the new queue is " + sqsQueueArn);
```

```
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Attach an IAM policy to the queue.");

        // Define the policy to use. Make sure that you change the REGION if you are
        // running this code
        // in a different region.
        String policy = """
        {
             "Statement": [
             {
                 "Effect": "Allow",
                         "Principal": {
                     "Service": "sns.amazonaws.com"
                 },
                 "Action": "sqs:SendMessage",
                         "Resource": "arn:aws:sqs:us-east-1:%s:%s",
                         "Condition": {
                     "ArnEquals": {
                         "aws:SourceArn": "arn:aws:sns:us-east-1:%s:%s"
                     }
                 }
             }
             ]
         }
        """.formatted(accountId, sqsQueueName, accountId, topicName);

        setQueueAttr(sqsClient, sqsQueueUrl, policy);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Subscribe to the SQS queue.");
        if (selectFIFO) {
            System.out.println(
                "If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.\n"
                    +
                    "For information about message filtering, see https://
docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n"
                    +
                    "For this example, you can filter messages by a \"tone\"
 attribute.");
```

```
            System.out.println("Would you like to filter messages for " +
sqsQueueName + "'s subscription to the topic "
                + topicName + "?  (y/n)");
            String filterAns = in.nextLine();
            if (filterAns.compareTo("y") == 0) {
                boolean moreAns = false;
                System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                while (!moreAns) {
                    System.out.println("Select a number or choose 0 to end.");
                    String ans = in.nextLine();
                    switch (ans) {
                        case "1":
                            filterList.add("cheerful");
                            break;
                        case "2":
                            filterList.add("funny");
                            break;
                        case "3":
                            filterList.add("serious");
                            break;
                        case "4":
                            filterList.add("sincere");
                            break;
                        default:
                            moreAns = true;
                            break;
                    }
                }
            }
        }
        subscriptionArn = subQueue(snsClient, topicArn, sqsQueueArn, filterList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Publish a message to the topic.");
        if (selectFIFO) {
            System.out.println("Would you like to add an attribute to this message?
(y/n)");
            String msgAns = in.nextLine();
```

```
            if (msgAns.compareTo("y") == 0) {
                System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                System.out.println("Select a number or choose 0 to end.");
                String ans = in.nextLine();
                switch (ans) {
                    case "1":
                        msgAttValue = "cheerful";
                        break;
                    case "2":
                        msgAttValue = "funny";
                        break;
                    case "3":
                        msgAttValue = "serious";
                        break;
                    default:
                        msgAttValue = "sincere";
                        break;
                }

                System.out.println("Selected value is " + msgAttValue);
            }
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessageFIFO(snsClient, message, topicArn, msgAttValue, duplication,
groupId, deduplicationID);

        } else {
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessage(snsClient, message, topicArn);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Display the message. Press any key to continue.");
        in.nextLine();
        messageList = receiveMessages(sqsClient, sqsQueueUrl, msgAttValue);
        for (Message mes : messageList) {
            System.out.println("Message Id: " + mes.messageId());
```

```
            System.out.println("Full Message: " + mes.body());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Delete the received message. Press any key to
continue.");
        in.nextLine();
        deleteMessages(sqsClient, sqsQueueUrl, messageList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Unsubscribe from the topic and delete the queue.
Press any key to continue.");
        in.nextLine();
        unSub(snsClient, subscriptionArn);
        deleteSQSQueue(sqsClient, sqsQueueName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete the topic. Press any key to continue.");
        in.nextLine();
        deleteSNSTopic(snsClient, topicArn);

        System.out.println(DASHES);
        System.out.println("The SNS/SQS workflow has completed successfully.");
        System.out.println(DASHES);
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                .topicArn(topicArn)
                .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("Status was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                .queueName(queueName)
                .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                .queueUrl(queueUrl)
                .build();

            sqsClient.deleteQueue(deleteQueueRequest);
            System.out.println(queueName + " was successfully deleted.");

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                .subscriptionArn(subscriptionArn)
                .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("Status was " + result.sdkHttpResponse().statusCode()
                + "\nSubscription was removed for " + request.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
        try {
            List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();
            for (Message msg : messages) {
                DeleteMessageBatchRequestEntry entry =
DeleteMessageBatchRequestEntry.builder()
```

```
                    .id(msg.messageId())
                    .build();

                entries.add(entry);
            }

            DeleteMessageBatchRequest deleteMessageBatchRequest =
DeleteMessageBatchRequest.builder()
                .queueUrl(queueUrl)
                .entries(entries)
                .build();

            sqsClient.deleteMessageBatch(deleteMessageBatchRequest);
            System.out.println("The batch delete of messages was successful");

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl, String msgAttValue) {
        try {
            if (msgAttValue.isEmpty()) {
                ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .maxNumberOfMessages(5)
                    .build();
                return sqsClient.receiveMessage(receiveMessageRequest).messages();
            } else {
                // We know there are filters on the message.
                ReceiveMessageRequest receiveRequest =
ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .messageAttributeNames(msgAttValue) // Include other message
attributes if needed.
                    .maxNumberOfMessages(5)
                    .build();

                return sqsClient.receiveMessage(receiveRequest).messages();
            }
```

```
        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static void pubMessage(SnsClient snsClient, String message, String
topicArn) {
        try {
            PublishRequest request = PublishRequest.builder()
                .message(message)
                .topicArn(topicArn)
                .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                .println(result.messageId() + " Message sent. Status is " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void pubMessageFIFO(SnsClient snsClient,
                                      String message,
                                      String topicArn,
                                      String msgAttValue,
                                      String duplication,
                                      String groupId,
                                      String deduplicationID) {

        try {
            PublishRequest request;
            // Means the user did not choose to use a message attribute.
            if (msgAttValue.isEmpty()) {
                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
```

```
                } else {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageDeduplicationId(deduplicationID)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                }

            } else {
                Map<String, MessageAttributeValue> messageAttributes = new
HashMap<>();
                messageAttributes.put(msgAttValue, MessageAttributeValue.builder()
                    .dataType("String")
                    .stringValue("true")
                    .build());

                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                        .message(message)
                        .messageGroupId(groupId)
                        .topicArn(topicArn)
                        .build();
                } else {
                    // Create a publish request with the message and attributes.
                    request = PublishRequest.builder()
                        .topicArn(topicArn)
                        .message(message)
                        .messageDeduplicationId(deduplicationID)
                        .messageGroupId(groupId)
                        .messageAttributes(messageAttributes)
                        .build();
                }
            }

            // Publish the message to the topic.
            PublishResponse result = snsClient.publish(request);
            System.out
                .println(result.messageId() + " Message sent. Status was " +
result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }

    // Subscribe to the SQS queue.
    public static String subQueue(SnsClient snsClient, String topicArn, String
queueArn, List<String> filterList) {
        try {
            SubscribeRequest request;
            if (filterList.isEmpty()) {
                // No filter subscription is added.
                request = SubscribeRequest.builder()
                    .protocol("sqs")
                    .endpoint(queueArn)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been subscribed
to the topic " + topicArn + "\n" +
                    "with the subscription ARN " + result.subscriptionArn());
                return result.subscriptionArn();
            } else {
                request = SubscribeRequest.builder()
                    .protocol("sqs")
                    .endpoint(queueArn)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been subscribed
to the topic " + topicArn + "\n" +
                    "with the subscription ARN " + result.subscriptionArn());

                String attributeName = "FilterPolicy";
                Gson gson = new Gson();
                String jsonString = "{\"tone\": []}";
                JsonObject jsonObject = gson.fromJson(jsonString, JsonObject.class);
                JsonArray toneArray = jsonObject.getAsJsonArray("tone");
                for (String value : filterList) {
                    toneArray.add(new JsonPrimitive(value));
                }
```

```
                String updatedJsonString = gson.toJson(jsonObject);
                System.out.println(updatedJsonString);
                SetSubscriptionAttributesRequest attRequest =
SetSubscriptionAttributesRequest.builder()
                    .subscriptionArn(result.subscriptionArn())
                    .attributeName(attributeName)
                    .attributeValue(updatedJsonString)
                    .build();

                snsClient.setSubscriptionAttributes(attRequest);
                return result.subscriptionArn();
            }

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    // Attach a policy to the queue.
    public static void setQueueAttr(SqsClient sqsClient, String queueUrl, String
policy) {
        try {
            Map<software.amazon.awssdk.services.sqs.model.QueueAttributeName,
String> attrMap = new HashMap<>();
            attrMap.put(QueueAttributeName.POLICY, policy);

            SetQueueAttributesRequest attributesRequest =
SetQueueAttributesRequest.builder()
                .queueUrl(queueUrl)
                .attributes(attrMap)
                .build();

            sqsClient.setQueueAttributes(attributesRequest);
            System.out.println("The policy has been successfully attached.");

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getSQSQueueAttrs(SqsClient sqsClient, String queueUrl) {
```

```
        // Specify the attributes to retrieve.
        List<QueueAttributeName> atts = new ArrayList<>();
        atts.add(QueueAttributeName.QUEUE_ARN);

        GetQueueAttributesRequest attributesRequest =
GetQueueAttributesRequest.builder()
            .queueUrl(queueUrl)
            .attributeNames(atts)
            .build();

        GetQueueAttributesResponse response =
sqsClient.getQueueAttributes(attributesRequest);
        Map<String, String> queueAtts = response.attributesAsStrings();
        for (Map.Entry<String, String> queueAtt : queueAtts.entrySet())
            return queueAtt.getValue();

        return "";
    }

    public static String createQueue(SqsClient sqsClient, String queueName, Boolean
selectFIFO) {
        try {
            System.out.println("\nCreate Queue");
            if (selectFIFO) {
                Map<QueueAttributeName, String> attrs = new HashMap<>();
                attrs.put(QueueAttributeName.FIFO_QUEUE, "true");
                CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .attributes(attrs)
                    .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            } else {
                CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
```

```
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            }

        } catch (SqsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                .name(topicName)
                .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createFIFO(SnsClient snsClient, String topicName, String
duplication) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = new HashMap<>();
            if (duplication.compareTo("n") == 0) {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "false");
            } else {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "true");
            }

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
```

```
                .name(topicName)
                .attributes(topicAttributes)
                .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            return response.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - 发布
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

发送和接收批量消息

以下代码示例展示了如何：

- 创建 Amazon SQS 队列。
- 将成批消息发送到队列。
- 从队列中接收成批消息。
- 从队列中删除成批消息。

适用于 Java 的 SDK 2.x

> **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。

如以下示例所示，可以使用 Amazon SQS，通过将两种不同的方法与 AWS SDK for Java 2.x结合使用来处理批量消息操作：

SendRecvBatch.java 使用显式批处理操作。您可以手动创建消息批次并直接调用 `sendMessageBatch()` 和 `deleteMessageBatch()`。还可以处理批量响应，包括任何失败的消息。这种方法使您可以完全控制批量大小和错误处理。但是，它需要更多的代码来管理批处理逻辑。

SimpleProducerConsumer.java 使用高级`SqsAsyncBatchManager`库进行自动请求批处理。您使用与标准客户端相同的方法签名进行单独的 `sendMessage()` 和 `deleteMessage()` 调用。SDK 会自动缓冲这些调用并将其作为批量操作发送。这种方法只需要最少的代码更改，同时提供批处理性能优势。

当您需要对批量组合和错误处理进行精细控制时，请使用显式批处理。如果您想以最少的代码更改来优化性能，请使用自动批处理。

SendRecvBatch.java-对消息使用显式批处理操作。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.BatchResultErrorEntry;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchResponse;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchResultEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.MessageAttributeValue;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.SendMessageBatchResponse;
```

```
import software.amazon.awssdk.services.sqs.model.SendMessageBatchResultEntry;
import software.amazon.awssdk.services.sqs.model.SqsException;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;


/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

/**
 * This code demonstrates basic message operations in Amazon Simple Queue Service
 (Amazon SQS).
 */

public class SendRecvBatch {
    private static final Logger LOGGER =
 LoggerFactory.getLogger(SendRecvBatch.class);
    private static final SqsClient sqsClient = SqsClient.create();


    public static void main(String[] args) {
        usageDemo();
    }
    /**
     * Send a batch of messages in a single request to an SQS queue.
     * This request may return overall success even when some messages were not
 sent.
     * The caller must inspect the Successful and Failed lists in the response and
     * resend any failed messages.
     *
     * @param queueUrl  The URL of the queue to receive the messages.
```

```
    * @param messages  The messages to send to the queue. Each message contains a
body and attributes.
    * @return The response from SQS that contains the list of successful and failed
messages.
    */
   public static SendMessageBatchResponse sendMessages(
           String queueUrl, List<MessageEntry> messages) {

       try {
           List<SendMessageBatchRequestEntry> entries = new ArrayList<>();

           for (int i = 0; i < messages.size(); i++) {
               MessageEntry message = messages.get(i);
               entries.add(SendMessageBatchRequestEntry.builder()
                       .id(String.valueOf(i))
                       .messageBody(message.getBody())
                       .messageAttributes(message.getAttributes())
                       .build());
           }

           SendMessageBatchRequest sendBatchRequest =
SendMessageBatchRequest.builder()
                   .queueUrl(queueUrl)
                   .entries(entries)
                   .build();

           SendMessageBatchResponse response =
sqsClient.sendMessageBatch(sendBatchRequest);

           if (!response.successful().isEmpty()) {
               for (SendMessageBatchResultEntry resultEntry :
response.successful()) {
                   LOGGER.info("Message sent: {}: {}", resultEntry.messageId(),

messages.get(Integer.parseInt(resultEntry.id())).getBody());
               }
           }

           if (!response.failed().isEmpty()) {
               for (BatchResultErrorEntry errorEntry : response.failed()) {
                   LOGGER.warn("Failed to send: {}: {}", errorEntry.id(),

messages.get(Integer.parseInt(errorEntry.id())).getBody());
               }
```

```
            }

            return response;

        } catch (SqsException e) {
            LOGGER.error("Send messages failed to queue: {}", queueUrl, e);
            throw e;
        }
    }

    /**
     * Receive a batch of messages in a single request from an SQS queue.
     *
     * @param queueUrl   The URL of the queue from which to receive messages.
     * @param maxNumber  The maximum number of messages to receive (capped at 10 by
SQS).
     *                   The actual number of messages received might be less.
     * @param waitTime   The maximum time to wait (in seconds) before returning.
When
     *                   this number is greater than zero, long polling is used.
This
     *                   can result in reduced costs and fewer false empty
responses.
     * @return The list of Message objects received. These each contain the body
     *         of the message and metadata and custom attributes.
     */
    public static List<Message> receiveMessages(String queueUrl, int maxNumber, int
waitTime) {
        try {
            ReceiveMessageRequest receiveRequest = ReceiveMessageRequest.builder()
                    .queueUrl(queueUrl)
                    .maxNumberOfMessages(maxNumber)
                    .waitTimeSeconds(waitTime)
                    .messageAttributeNames("All")
                    .build();

            List<Message> messages =
sqsClient.receiveMessage(receiveRequest).messages();

            for (Message message : messages) {
                LOGGER.info("Received message: {}: {}", message.messageId(),
message.body());
            }
```

```
            return messages;

        } catch (SqsException e) {
            LOGGER.error("Couldn't receive messages from queue: {}", queueUrl, e);
            throw e;
        }
    }

    /**
     * Delete a batch of messages from a queue in a single request.
     *
     * @param queueUrl  The URL of the queue from which to delete the messages.
     * @param messages  The list of messages to delete.
     * @return The response from SQS that contains the list of successful and failed
     *         message deletions.
     */
    public static DeleteMessageBatchResponse deleteMessages(String queueUrl,
List<Message> messages) {
        try {
            List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();

            for (int i = 0; i < messages.size(); i++) {
                entries.add(DeleteMessageBatchRequestEntry.builder()
                        .id(String.valueOf(i))
                        .receiptHandle(messages.get(i).receiptHandle())
                        .build());
            }

            DeleteMessageBatchRequest deleteRequest =
DeleteMessageBatchRequest.builder()
                    .queueUrl(queueUrl)
                    .entries(entries)
                    .build();

            DeleteMessageBatchResponse response =
sqsClient.deleteMessageBatch(deleteRequest);

            if (!response.successful().isEmpty()) {
                for (DeleteMessageBatchResultEntry resultEntry :
response.successful()) {
                    LOGGER.info("Deleted {}",
messages.get(Integer.parseInt(resultEntry.id())).receiptHandle());
                }
            }
```

```
            if (!response.failed().isEmpty()) {
                for (BatchResultErrorEntry errorEntry : response.failed()) {
                    LOGGER.warn("Could not delete {}",
messages.get(Integer.parseInt(errorEntry.id())).receiptHandle());
                }
            }

            return response;

        } catch (SqsException e) {
            LOGGER.error("Couldn't delete messages from queue {}", queueUrl, e);
            throw e;
        }
    }

    /**
     * Helper class to represent a message with body and attributes.
     */
    public static class MessageEntry {
        private final String body;
        private final Map<String, MessageAttributeValue> attributes;

        public MessageEntry(String body, Map<String, MessageAttributeValue>
attributes) {
            this.body = body;
            this.attributes = attributes != null ? attributes : new HashMap<>();
        }

        public String getBody() {
            return body;
        }

        public Map<String, MessageAttributeValue> getAttributes() {
            return attributes;
        }
    }

    /**
     * Shows how to:
     * * Read the lines from a file and send the lines in
     *   batches of 10 as messages to a queue.
     * * Receive the messages in batches until the queue is empty.
     * * Reassemble the lines of the file and verify they match the original file.
```

```
     */
    public static void usageDemo() {
        LOGGER.info("-".repeat(88));
        LOGGER.info("Welcome to the Amazon Simple Queue Service (Amazon SQS)
demo!");
        LOGGER.info("-".repeat(88));

        String queueUrl = null;
        try {
            // Create a queue for the demo.
            String queueName = "sqs-usage-demo-message-wrapper-" +
System.currentTimeMillis();
            CreateQueueRequest createRequest = CreateQueueRequest.builder()
                    .queueName(queueName)
                    .build();
            queueUrl = sqsClient.createQueue(createRequest).queueUrl();
            LOGGER.info("Created queue: {}", queueUrl);

            try (InputStream inputStream =
SendRecvBatch.class.getResourceAsStream("/log4j2.xml");
                 BufferedReader reader = new BufferedReader(new
InputStreamReader(inputStream))) {

                List<String> lines = reader.lines().toList();

                // Send file lines in batches.
                int batchSize = 10;
                LOGGER.info("Sending file lines in batches of {} as messages.",
batchSize);

                for (int i = 0; i < lines.size(); i += batchSize) {
                    List<MessageEntry> messageBatch = new ArrayList<>();

                    for (int j = i; j < Math.min(i + batchSize, lines.size()); j++)
{
                        String line = lines.get(j);
                        if (line == null || line.trim().isEmpty()) {
                            continue; // Skip empty lines.
                        }

                        Map<String, MessageAttributeValue> attributes = new
HashMap<>();
                        attributes.put("line", MessageAttributeValue.builder()
                                .dataType("String")
```

```
                                .stringValue(String.valueOf(j))
                                .build());

                        messageBatch.add(new MessageEntry(lines.get(j),
attributes));
                    }

                    sendMessages(queueUrl, messageBatch);
                    System.out.print(".");
                    System.out.flush();
                }

                LOGGER.info("\nDone. Sent {} messages.", lines.size());

                // Receive and process messages.
                LOGGER.info("Receiving, handling, and deleting messages in batches
of {}.", batchSize);
                String[] receivedLines = new String[lines.size()];
                boolean moreMessages = true;

                while (moreMessages) {
                    List<Message> receivedMessages = receiveMessages(queueUrl,
batchSize, 5);

                    for (Message message : receivedMessages) {
                        int lineNumber =
Integer.parseInt(message.messageAttributes().get("line").stringValue());
                        receivedLines[lineNumber] = message.body();
                    }

                    if (!receivedMessages.isEmpty()) {
                        deleteMessages(queueUrl, receivedMessages);
                    } else {
                        moreMessages = false;
                    }
                }

                LOGGER.info("\nDone.");

                // Verify that all lines were received correctly.
                boolean allLinesMatch = true;
                for (int i = 0; i < lines.size(); i++) {
                    String originalLine = lines.get(i);
```

```
                    String receivedLine = receivedLines[i] == null ? "" :
receivedLines[i];

                    if (!originalLine.equals(receivedLine)) {
                        allLinesMatch = false;
                        break;
                    }
                }

                if (allLinesMatch) {
                    LOGGER.info("Successfully reassembled all file lines!");
                } else {
                    LOGGER.info("Uh oh, some lines were missed!");
                }
            }
        } catch (SqsException e) {
            LOGGER.error("SQS operation failed", e);
        } catch (RuntimeException | IOException e) {
            LOGGER.error("Unexpected runtime error during demo", e);
        } finally {
            // Clean up by deleting the queue if it was created.
            if (queueUrl != null) {
                try {
                    DeleteQueueRequest deleteQueueRequest =
DeleteQueueRequest.builder()
                            .queueUrl(queueUrl)
                            .build();
                    sqsClient.deleteQueue(deleteQueueRequest);
                    LOGGER.info("Deleted queue: {}", queueUrl);
                } catch (SqsException e) {
                    LOGGER.error("Failed to delete queue: {}", queueUrl, e);
                }
            }
        }

        LOGGER.info("Thanks for watching!");
        LOGGER.info("-".repeat(88));
    }
}
```

SimpleProducerConsumer.java-使用自动批处理消息。

```
package com.example.sqs;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.sqs.SqsAsyncClient;
import software.amazon.awssdk.services.sqs.batchmanager.SqsAsyncBatchManager;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
import software.amazon.awssdk.services.sqs.model.SendMessageResponse;
import software.amazon.awssdk.core.exception.SdkException;

import java.math.BigInteger;
import java.util.List;
import java.util.Random;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.atomic.AtomicBoolean;
import java.util.concurrent.atomic.AtomicInteger;

/**
 * Demonstrates the AWS SDK for Java 2.x Automatic Request Batching API for Amazon
 SQS.
 *
 * This example showcases the high-level SqsAsyncBatchManager library that provides
 * efficient batching and buffering for SQS operations. The batch manager offers
 * methods that directly mirror SqsAsyncClient methods—sendMessage,
 changeMessageVisibility,
 * deleteMessage, and receiveMessage—making it a drop-in replacement with minimal
 code changes.
 *
 * Key features of the SqsAsyncBatchManager:
 * - Automatic batching: The SDK automatically buffers individual requests and sends
 them
 *   as batches when maxBatchSize (default: 10) or sendRequestFrequency (default:
 200ms)
 *   thresholds are reached
```

```
 * - Familiar API: Method signatures match SqsAsyncClient exactly, requiring no
 learning curve
 * - Background optimization: The batch manager maintains internal buffers and
 handles
 *   batching logic transparently
 * - Asynchronous operations: All methods return CompletableFuture for non-blocking
 execution
 *
 * Performance benefits demonstrated:
 * - Reduced API calls: Multiple individual requests are consolidated into single
 batch operations
 * - Lower costs: Fewer API calls result in reduced SQS charges
 * - Higher throughput: Batch operations process more messages per second
 * - Efficient resource utilization: Fewer network round trips and better connection
 reuse
 *
 * This example compares:
 * 1. Single-message operations using SqsAsyncClient directly
 * 2. Batch operations using SqsAsyncBatchManager with identical method calls
 *
 * Usage patterns:
 * - Set batch size to 1 to use SqsAsyncClient for baseline performance measurement
 * - Set batch size > 1 to use SqsAsyncBatchManager for optimized batch processing
 * - Monitor real-time throughput metrics to observe performance improvements
 *
 * Prerequisites:
 * - AWS SDK for Java 2.x version 2.28.0 or later
 * - An existing SQS queue
 * - Valid AWS credentials configured
 *
 * The program displays real-time metrics showing the dramatic performance
 difference
 * between individual operations and automatic batching.
 */
public class SimpleProducerConsumer {

    // The maximum runtime of the program.
    private final static int MAX_RUNTIME_MINUTES = 60;
    private final static Logger log =
 LoggerFactory.getLogger(SimpleProducerConsumer.class);

    /**
     * Runs the SQS batching demonstration with user-configured parameters.
     *
```

```
     * Prompts for queue name, thread counts, batch size, message size, and runtime.
     * Creates producer and consumer threads to demonstrate batching performance.
     *
     * @param args command line arguments (not used)
     * @throws InterruptedException if thread operations are interrupted
     */
    public static void main(String[] args) throws InterruptedException {

        final Scanner input = new Scanner(System.in);

        System.out.print("Enter the queue name: ");
        final String queueName = input.nextLine();

        System.out.print("Enter the number of producers: ");
        final int producerCount = input.nextInt();

        System.out.print("Enter the number of consumers: ");
        final int consumerCount = input.nextInt();

        System.out.print("Enter the number of messages per batch: ");
        final int batchSize = input.nextInt();

        System.out.print("Enter the message size in bytes: ");
        final int messageSizeByte = input.nextInt();

        System.out.print("Enter the run time in minutes: ");
        final int runTimeMinutes = input.nextInt();

        // Create SQS async client and batch manager for all operations.
        // The SqsAsyncBatchManager is created from the SqsAsyncClient using the
        // batchManager() factory method, which provides default batching
configuration.
        // This high-level library automatically handles request buffering and
batching
        // while maintaining the same method signatures as SqsAsyncClient.
        final SqsAsyncClient sqsAsyncClient = SqsAsyncClient.create();
        final SqsAsyncBatchManager batchManager = sqsAsyncClient.batchManager();

        final String queueUrl =
sqsAsyncClient.getQueueUrl(GetQueueUrlRequest.builder()
                .queueName(queueName)
                .build()).join().queueUrl();

        // The flag used to stop producer, consumer, and monitor threads.
```

```
        final AtomicBoolean stop = new AtomicBoolean(false);

        // Start the producers.
        final AtomicInteger producedCount = new AtomicInteger();
        final Thread[] producers = new Thread[producerCount];
        for (int i = 0; i < producerCount; i++) {
            if (batchSize == 1) {
                producers[i] = new Producer(sqsAsyncClient, queueUrl,
messageSizeByte,
                        producedCount, stop);
            } else {
                producers[i] = new BatchProducer(batchManager, queueUrl, batchSize,
                        messageSizeByte, producedCount, stop);
            }
            producers[i].start();
        }

        // Start the consumers.
        final AtomicInteger consumedCount = new AtomicInteger();
        final Thread[] consumers = new Thread[consumerCount];
        for (int i = 0; i < consumerCount; i++) {
            if (batchSize == 1) {
                consumers[i] = new Consumer(sqsAsyncClient, queueUrl, consumedCount,
stop);
            } else {
                consumers[i] = new BatchConsumer(batchManager, queueUrl, batchSize,
                        consumedCount, stop);
            }
            consumers[i].start();
        }

        // Start the monitor thread.
        final Thread monitor = new Monitor(producedCount, consumedCount, stop);
        monitor.start();

        // Wait for the specified amount of time then stop.
        Thread.sleep(TimeUnit.MINUTES.toMillis(Math.min(runTimeMinutes,
                MAX_RUNTIME_MINUTES)));
        stop.set(true);

        // Join all threads.
        for (int i = 0; i < producerCount; i++) {
            producers[i].join();
        }
```

```
        for (int i = 0; i < consumerCount; i++) {
            consumers[i].join();
        }

        monitor.interrupt();
        monitor.join();

        // Close resources
        batchManager.close();
        sqsAsyncClient.close();
    }

    /**
     * Creates a random string of approximately the specified size in bytes.
     *
     * @param sizeByte the target size in bytes for the generated string
     * @return a random string encoded in base-32
     */
    private static String makeRandomString(int sizeByte) {
        final byte[] bs = new byte[(int) Math.ceil(sizeByte * 5 / 8)];
        new Random().nextBytes(bs);
        bs[0] = (byte) ((bs[0] | 64) & 127);
        return new BigInteger(bs).toString(32);
    }

    /**
     * Sends messages individually using SqsAsyncClient for baseline performance
measurement.
     *
     * This producer demonstrates traditional single-message operations without
batching.
     * Each sendMessage() call results in a separate API request to SQS, providing
     * a performance baseline for comparison with the batch operations.
     *
     * The sendMessage() method signature is identical to
SqsAsyncBatchManager.sendMessage(),
     * showing how the high-level batching library maintains API compatibility while
     * adding automatic optimization behind the scenes.
     */
    private static class Producer extends Thread {
        final SqsAsyncClient sqsAsyncClient;
        final String queueUrl;
        final AtomicInteger producedCount;
```

```
        final AtomicBoolean stop;
        final String theMessage;

        /**
         * Creates a producer thread for single-message operations.
         *
         * @param sqsAsyncClient the SQS client for sending messages
         * @param queueUrl the URL of the target queue
         * @param messageSizeByte the size of messages to generate
         * @param producedCount shared counter for tracking sent messages
         * @param stop shared flag to signal thread termination
         */
        Producer(SqsAsyncClient sqsAsyncClient, String queueUrl, int
messageSizeByte,
                 AtomicInteger producedCount, AtomicBoolean stop) {
            this.sqsAsyncClient = sqsAsyncClient;
            this.queueUrl = queueUrl;
            this.producedCount = producedCount;
            this.stop = stop;
            this.theMessage = makeRandomString(messageSizeByte);
        }

        /**
         * Continuously sends messages until the stop flag is set.
         *
         * Uses SqsAsyncClient.sendMessage() directly, resulting in one API call per
message.
         * This approach provides baseline performance metrics for comparison with
batching.
         * Each call blocks until the individual message is sent, demonstrating
traditional
         * one-request-per-operation behavior.
         */
        public void run() {
            try {
                while (!stop.get()) {
                    sqsAsyncClient.sendMessage(SendMessageRequest.builder()
                            .queueUrl(queueUrl)
                            .messageBody(theMessage)
                            .build()).join();
                    producedCount.incrementAndGet();
                }
            } catch (SdkException | java.util.concurrent.CompletionException e) {
```

```
                // Handle both SdkException and CompletionException from async
operations.
                // If this unlikely condition occurs, stop.
                log.error("Producer: " + e.getMessage());
                System.exit(1);
            }
        }
    }

    /**
     * Sends messages using SqsAsyncBatchManager for automatic request batching and
optimization.
     *
     * This producer demonstrates the AWS SDK for Java 2.x high-level batching
library.
     * The SqsAsyncBatchManager automatically buffers individual sendMessage() calls
and
     * sends them as batches when thresholds are reached:
     * - maxBatchSize: Maximum 10 messages per batch (default)
     * - sendRequestFrequency: 200ms timeout before sending partial batches
(default)
     *
     * Key advantages of the batching approach:
     * - Identical API: batchManager.sendMessage() has the same signature as
sqsAsyncClient.sendMessage()
     * - Automatic optimization: No code changes needed to benefit from batching
     * - Transparent buffering: The SDK handles batching logic internally
     * - Reduced API calls: Multiple messages sent in single batch requests
     * - Lower costs: Fewer API calls result in reduced SQS charges
     * - Higher throughput: Batch operations process significantly more messages per
second
     */
    private static class BatchProducer extends Thread {
        final SqsAsyncBatchManager batchManager;
        final String queueUrl;
        final int batchSize;
        final AtomicInteger producedCount;
        final AtomicBoolean stop;
        final String theMessage;

        /**
         * Creates a producer thread for batch operations.
         *
         * @param batchManager the batch manager for efficient message sending
```

```
         * @param queueUrl the URL of the target queue
         * @param batchSize the number of messages to send per batch
         * @param messageSizeByte the size of messages to generate
         * @param producedCount shared counter for tracking sent messages
         * @param stop shared flag to signal thread termination
         */
        BatchProducer(SqsAsyncBatchManager batchManager, String queueUrl, int
batchSize,
                      int messageSizeByte, AtomicInteger producedCount,
                      AtomicBoolean stop) {
            this.batchManager = batchManager;
            this.queueUrl = queueUrl;
            this.batchSize = batchSize;
            this.producedCount = producedCount;
            this.stop = stop;
            this.theMessage = makeRandomString(messageSizeByte);
        }

        /**
         * Continuously sends batches of messages using the high-level batching
library.
         *
         * Notice how batchManager.sendMessage() uses the exact same method
signature
         * and request builder pattern as SqsAsyncClient.sendMessage(). This
demonstrates
         * the drop-in replacement capability of the SqsAsyncBatchManager.
         *
         * The SDK automatically:
         * - Buffers individual sendMessage() calls internally
         * - Groups them into batch requests when thresholds are met
         * - Sends SendMessageBatchRequest operations to SQS
         * - Returns individual CompletableFuture responses for each message
         *
         * This transparent batching provides significant performance improvements
         * without requiring changes to application logic or error handling
patterns.
         */
        public void run() {
            try {
                while (!stop.get()) {
                    // Send multiple messages using the high-level batch manager.
                    // Each batchManager.sendMessage() call uses identical syntax to
```

```
                    // sqsAsyncClient.sendMessage(), demonstrating API
compatibility.
                    // The SDK automatically buffers these calls and sends them as
                    // batch operations when maxBatchSize (10) or
sendRequestFrequency (200ms)
                    // thresholds are reached, significantly improving throughput.
                    for (int i = 0; i < batchSize; i++) {
                        CompletableFuture<SendMessageResponse> future =
batchManager.sendMessage(
                                SendMessageRequest.builder()
                                        .queueUrl(queueUrl)
                                        .messageBody(theMessage)
                                        .build());

                        // Handle the response asynchronously
                        future.whenComplete((response, throwable) -> {
                            if (throwable == null) {
                                producedCount.incrementAndGet();
                            } else if (!(throwable instanceof
java.util.concurrent.CancellationException) &&
                                    !(throwable.getMessage() != null &&
throwable.getMessage().contains("executor not accepting a task"))) {
                                log.error("BatchProducer: Failed to send message",
throwable);
                            }
                            // Ignore CancellationException and executor shutdown
errors - expected during shutdown
                        });
                    }

                    // Small delay to allow batching to occur
                    Thread.sleep(10);
                }
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                log.error("BatchProducer interrupted: " + e.getMessage());
            } catch (SdkException | java.util.concurrent.CompletionException e) {
                log.error("BatchProducer: " + e.getMessage());
                System.exit(1);
            }
        }
    }

    /**
```

```
    * Receives and deletes messages individually using SqsAsyncClient for baseline
measurement.
    *
    * This consumer demonstrates traditional single-message operations without
batching.
    * Each receiveMessage() and deleteMessage() call results in separate API
requests,
    * providing a performance baseline for comparison with batch operations.
    *
    * The method signatures are identical to SqsAsyncBatchManager methods:
    * - receiveMessage() matches batchManager.receiveMessage()
    * - deleteMessage() matches batchManager.deleteMessage()
    *
    * This API consistency allows easy migration to the high-level batching
library.
    */
   private static class Consumer extends Thread {
       final SqsAsyncClient sqsAsyncClient;
       final String queueUrl;
       final AtomicInteger consumedCount;
       final AtomicBoolean stop;

       /**
        * Creates a consumer thread for single-message operations.
        *
        * @param sqsAsyncClient the SQS client for receiving messages
        * @param queueUrl the URL of the source queue
        * @param consumedCount shared counter for tracking processed messages
        * @param stop shared flag to signal thread termination
        */
       Consumer(SqsAsyncClient sqsAsyncClient, String queueUrl, AtomicInteger
consumedCount,
                AtomicBoolean stop) {
           this.sqsAsyncClient = sqsAsyncClient;
           this.queueUrl = queueUrl;
           this.consumedCount = consumedCount;
           this.stop = stop;
       }

       /**
        * Continuously receives and deletes messages using traditional single-
request operations.
        *
        * Uses SqsAsyncClient methods directly:
```

```
         * - receiveMessage(): One API call per receive operation
         * - deleteMessage(): One API call per delete operation
         *
         * This approach demonstrates the baseline performance without batching
optimization.
         * Compare these method calls with the identical signatures used in
BatchConsumer
         * to see how the high-level batching library maintains API compatibility.
         */
        public void run() {
            try {
                while (!stop.get()) {
                    try {
                        final ReceiveMessageResponse result =
sqsAsyncClient.receiveMessage(
                                ReceiveMessageRequest.builder()
                                        .queueUrl(queueUrl)
                                        .build()).join();

                        if (!result.messages().isEmpty()) {
                            final Message m = result.messages().get(0);
                            // Note: deleteMessage() signature identical to
batchManager.deleteMessage()

sqsAsyncClient.deleteMessage(DeleteMessageRequest.builder()
                                    .queueUrl(queueUrl)
                                    .receiptHandle(m.receiptHandle())
                                    .build()).join();
                            consumedCount.incrementAndGet();
                        }
                    } catch (SdkException | java.util.concurrent.CompletionException
e) {
                        log.error(e.getMessage());
                    }
                }
            } catch (SdkException | java.util.concurrent.CompletionException e) {
                // Handle both SdkException and CompletionException from async
operations.
                // If this unlikely condition occurs, stop.
                log.error("Consumer: " + e.getMessage());
                System.exit(1);
            }
        }
    }
```

```
   /**
    * Receives and deletes messages using SqsAsyncBatchManager for automatic
optimization.
    *
    * This consumer demonstrates the AWS SDK for Java 2.x high-level batching
library
    * for message consumption. The SqsAsyncBatchManager provides two key
optimizations:
    *
    * 1. Receive optimization: Maintains an internal buffer of messages fetched in
the
    *    background, so receiveMessage() calls return immediately from the buffer
    * 2. Delete batching: Automatically buffers deleteMessage() calls and sends
them
    *    as DeleteMessageBatchRequest operations when thresholds are reached
    *
    * Key features:
    * - Identical API: receiveMessage() and deleteMessage() have the same
signatures
    *   as SqsAsyncClient methods, making this a true drop-in replacement
    * - Background fetching: The batch manager continuously fetches messages to
keep
    *   the internal buffer populated, reducing receive latency
    * - Automatic delete batching: Individual deleteMessage() calls are buffered
and
    *   sent as batch operations (up to 10 per batch, 200ms frequency)
    * - Transparent optimization: No application logic changes needed to benefit
    *
    * Performance benefits:
    * - Reduced API calls through automatic batching of delete operations
    * - Lower latency for receives due to background message buffering
    * - Higher overall throughput with fewer network round trips
    */
   private static class BatchConsumer extends Thread {
       final SqsAsyncBatchManager batchManager;
       final String queueUrl;
       final int batchSize;
       final AtomicInteger consumedCount;
       final AtomicBoolean stop;

       /**
        * Creates a consumer thread for batch operations.
        *
```

```
         * @param batchManager the batch manager for efficient message processing
         * @param queueUrl the URL of the source queue
         * @param batchSize the maximum number of messages to receive per batch
         * @param consumedCount shared counter for tracking processed messages
         * @param stop shared flag to signal thread termination
         */
        BatchConsumer(SqsAsyncBatchManager batchManager, String queueUrl, int
batchSize,
                      AtomicInteger consumedCount, AtomicBoolean stop) {
            this.batchManager = batchManager;
            this.queueUrl = queueUrl;
            this.batchSize = batchSize;
            this.consumedCount = consumedCount;
            this.stop = stop;
        }

        /**
         * Continuously receives and deletes messages using the high-level batching
library.
         *
         * Demonstrates the key advantage of SqsAsyncBatchManager: identical method
signatures
         * with automatic optimization. Notice how:
         *
         * - batchManager.receiveMessage() uses the same syntax as
sqsAsyncClient.receiveMessage()
         * - batchManager.deleteMessage() uses the same syntax as
sqsAsyncClient.deleteMessage()
         *
         * Behind the scenes, the batch manager:
         * 1. Maintains an internal message buffer populated by background fetching
         * 2. Returns messages immediately from the buffer (reduced latency)
         * 3. Automatically batches deleteMessage() calls into
DeleteMessageBatchRequest operations
         * 4. Sends batch deletes when maxBatchSize (10) or sendRequestFrequency
(200ms) is reached
         *
         * This provides significant performance improvements with zero code changes
         * compared to traditional SqsAsyncClient usage patterns.
         */
        public void run() {
            try {
                while (!stop.get()) {
                    // Receive messages using the high-level batch manager.
```

```
                    // This call uses identical syntax to
sqsAsyncClient.receiveMessage()
                    // but benefits from internal message buffering for improved
performance.
                    final ReceiveMessageResponse result =
batchManager.receiveMessage(
                            ReceiveMessageRequest.builder()
                                    .queueUrl(queueUrl)
                                    .maxNumberOfMessages(Math.min(batchSize, 10))
                                    .build()).join();

                    if (!result.messages().isEmpty()) {
                        final List<Message> messages = result.messages();

                        // Delete messages using the batch manager.
                        // Each deleteMessage() call uses identical syntax to
SqsAsyncClient
                        // but the SDK automatically buffers these calls and sends
them
                        // as DeleteMessageBatchRequest operations for optimal
performance.
                        for (Message message : messages) {
                            CompletableFuture<DeleteMessageResponse> future =
batchManager.deleteMessage(
                                    DeleteMessageRequest.builder()
                                            .queueUrl(queueUrl)
                                            .receiptHandle(message.receiptHandle())
                                            .build());

                            future.whenComplete((response, throwable) -> {
                                if (throwable == null) {
                                    consumedCount.incrementAndGet();
                                } else if (!(throwable instanceof
java.util.concurrent.CancellationException) &&
                                        !(throwable.getMessage() != null &&
throwable.getMessage().contains("executor not accepting a task"))) {
                                    log.error("BatchConsumer: Failed to delete
message", throwable);
                                }
                                // Ignore CancellationException and executor
shutdown errors - expected during shutdown
                            });
                        }
                    }
```

```
                    // Small delay to prevent tight polling
                    Thread.sleep(10);
                }
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                log.error("BatchConsumer interrupted: " + e.getMessage());
            } catch (SdkException | java.util.concurrent.CompletionException e) {
                // Handle both SdkException and CompletionException from async
operations.
                // If this unlikely condition occurs, stop.
                log.error("BatchConsumer: " + e.getMessage());
                System.exit(1);
            }
        }
    }

    /**
     * Displays real-time throughput statistics every second.
     *
     * This thread logs the current count of produced and consumed messages
     * to help you monitor the performance comparison.
     */
    private static class Monitor extends Thread {
        private final AtomicInteger producedCount;
        private final AtomicInteger consumedCount;
        private final AtomicBoolean stop;

        /**
         * Creates a monitoring thread that displays throughput statistics.
         *
         * @param producedCount shared counter for messages sent
         * @param consumedCount shared counter for messages processed
         * @param stop shared flag to signal thread termination
         */
        Monitor(AtomicInteger producedCount, AtomicInteger consumedCount,
                AtomicBoolean stop) {
            this.producedCount = producedCount;
            this.consumedCount = consumedCount;
            this.stop = stop;
        }

        /**
         * Logs throughput statistics every second until stopped.
```

```
         *
         * Displays the current count of produced and consumed messages
         * to help monitor the performance comparison between batching strategies.
         */
        public void run() {
            try {
                while (!stop.get()) {
                    Thread.sleep(1000);
                    log.info("produced messages = " + producedCount.get()
                            + ", consumed messages = " + consumedCount.get());
                }
            } catch (InterruptedException e) {
                // Allow the thread to exit.
            }
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - CreateQueue
  - DeleteMessage
  - DeleteMessageBatch
  - DeleteQueue
  - ReceiveMessage
  - SendMessage
  - SendMessageBatch

使用 Amazon SQS Java Messaging Library 处理 JMS 接口

以下代码示例演示如何使用 Amazon SQS Java Messaging Library 处理 JMS 接口。

适用于 Java 的 SDK 2.x

> **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。

以下示例处理标准 Amazon SQS 队列，包括：

- 发送文本消息。
- 同步接收消息。
- 异步接收消息。
- 使用 CLIENT_ACKNOWLEDGE 模式接收消息。
- 使用 UNORDERED_ACKNOWLEDGE 模式接收消息。
- 使用 Spring 注入依赖关系。
- 一个实用程序类，提供由其它示例使用的常用方法。

有关将 JMS 与 Amazon SQS 结合使用的更多信息，请参阅 Amazon SQS Developer Guide。

发送文本消息。

```
   /**
    * This method establishes a connection to a standard Amazon SQS queue using the
Amazon SQS
    * Java Messaging Library and sends text messages to it. It uses JMS (Java
Message Service) API
    * with automatic acknowledgment mode to ensure reliable message delivery, and
automatically
    * manages all messaging resources.
    *
    * @throws JMSException If there is a problem connecting to or sending messages
to the queue
    */
   public static void doSendTextMessage() throws JMSException {
       // Create a connection factory.
       SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
               new ProviderConfiguration(),
               SqsClient.create()
       );

       // Create the connection in a try-with-resources statement so that it's
closed automatically.
       try (SQSConnection connection = connectionFactory.createConnection()) {

           // Create the queue if needed.
           SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);

           // Create a session that uses the JMS auto-acknowledge mode.
```

```
            Session session = connection.createSession(false,
Session.AUTO_ACKNOWLEDGE);
            MessageProducer producer =
session.createProducer(session.createQueue(QUEUE_NAME));

            createAndSendMessages(session, producer);
        } // The connection closes automatically. This also closes the session.
        LOGGER.info("Connection closed");
    }

    /**
     * This method reads text input from the keyboard and sends each line as a
separate message
     * to a standard Amazon SQS queue using the Amazon SQS Java Messaging Library.
It continues
     * to accept input until the user enters an empty line, using JMS (Java Message
Service) API to
     * handle the message delivery.
     *
     * @param session The JMS session used to create messages
     * @param producer The JMS message producer used to send messages to the queue
     */
    private static void createAndSendMessages(Session session, MessageProducer
producer) {
        BufferedReader inputReader = new BufferedReader(
                new InputStreamReader(System.in, Charset.defaultCharset()));

        try {
            String input;
            while (true) {
                LOGGER.info("Enter message to send (leave empty to exit): ");
                input = inputReader.readLine();
                if (input == null || input.isEmpty()) break;

                TextMessage message = session.createTextMessage(input);
                producer.send(message);
                LOGGER.info("Send message {}", message.getJMSMessageID());
            }
        } catch (EOFException e) {
            // Just return on EOF
        } catch (IOException e) {
            LOGGER.error("Failed reading input: {}", e.getMessage(), e);
        } catch (JMSException e) {
            LOGGER.error("Failed sending message: {}", e.getMessage(), e);
```

```
        }
    }
```

同步接收消息。

```
    /**
     * This method receives messages from a standard Amazon SQS queue using the
Amazon SQS Java
     * Messaging Library. It creates a connection to the queue using JMS (Java
Message Service),
     * waits for messages to arrive, and processes them one at a time. The method
handles all
     * necessary setup and cleanup of messaging resources.
     *
     * @throws JMSException If there is a problem connecting to or receiving
messages from the queue
     */
    public static void doReceiveMessageSync() throws JMSException {
        // Create a connection factory.
        SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
                new ProviderConfiguration(),
                SqsClient.create()
        );

        // Create a connection.
        try (SQSConnection connection = connectionFactory.createConnection() ) {

            // Create the queue if needed.
            SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);

            // Create a session.
            Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);
            MessageConsumer consumer =
session.createConsumer(session.createQueue(QUEUE_NAME));

            connection.start();

            receiveMessages(consumer);
        }  // The connection closes automatically. This also closes the session.
        LOGGER.info("Connection closed");
```

```
    }

    /**
     * This method continuously checks for new messages from a standard Amazon SQS
queue using
     * the Amazon SQS Java Messaging Library. It waits up to 20 seconds for each
message, processes
     * it using JMS (Java Message Service), and confirms receipt. The method stops
checking for
     * messages after 20 seconds of no activity.
     *
     * @param consumer The JMS message consumer that receives messages from the
queue
     */
    private static void receiveMessages(MessageConsumer consumer) {
        try {
            while (true) {
                LOGGER.info("Waiting for messages...");
                // Wait 1 minute for a message
                Message message =
consumer.receive(Duration.ofSeconds(20).toMillis());
                if (message == null) {
                    LOGGER.info("Shutting down after 20 seconds of silence.");
                    break;
                }
                SqsJmsExampleUtils.handleMessage(message);
                message.acknowledge();
                LOGGER.info("Acknowledged message {}", message.getJMSMessageID());
            }
        } catch (JMSException e) {
            LOGGER.error("Error receiving from SQS: {}", e.getMessage(), e);
        }
    }
```

异步接收消息。

```
    /**
     * This method sets up automatic message handling for a standard Amazon SQS
queue using the
     * Amazon SQS Java Messaging Library. It creates a listener that processes
messages as soon
```

```
    * as they arrive using JMS (Java Message Service), runs for 5 seconds, then
cleans up all
    * messaging resources.
    *
    * @throws JMSException If there is a problem connecting to or receiving
messages from the queue
    */
   public static void doReceiveMessageAsync() throws JMSException {
       // Create a connection factory.
       SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
               new ProviderConfiguration(),
               SqsClient.create()
       );

       // Create a connection.
       try (SQSConnection connection = connectionFactory.createConnection() ) {

           // Create the queue if needed.
           SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);

           // Create a session.
           Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);

           try {
               // Create a consumer for the queue.
               MessageConsumer consumer =
session.createConsumer(session.createQueue(QUEUE_NAME));
               // Provide an implementation of the MessageListener interface, which
has a single 'onMessage' method.
               // We use a lambda expression for the implementation.
               consumer.setMessageListener(message -> {
                   try {
                       SqsJmsExampleUtils.handleMessage(message);
                       message.acknowledge();
                   } catch (JMSException e) {
                       LOGGER.error("Error processing message: {}",
e.getMessage());
                   }
               });
               // Start receiving incoming messages.
               connection.start();
               LOGGER.info("Waiting for messages...");
```

```
            } catch (JMSException e) {
                throw new RuntimeException(e);
            }
            try {
                Thread.sleep(5000);
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
        }  // The connection closes automatically. This also closes the session.
        LOGGER.info( "Connection closed" );
    }
```

使用 CLIENT_ACKNOWLEDGE 模式接收消息。

```
    /**
     * This method demonstrates how message acknowledgment affects message
processing in a standard
     * Amazon SQS queue using the Amazon SQS Java Messaging Library. It sends
messages to the queue,
     * then shows how JMS (Java Message Service) client acknowledgment mode handles
both explicit
     * and implicit message confirmations, including how acknowledging one message
can automatically
     * acknowledge previous messages.
     *
     * @throws JMSException If there is a problem with the messaging operations
     */
    public static void doReceiveMessagesSyncClientAcknowledge() throws JMSException
{
        // Create a connection factory.
        SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
                new ProviderConfiguration(),
                SqsClient.create()
        );

        // Create the connection in a try-with-resources statement so that it's
closed automatically.
        try (SQSConnection connection = connectionFactory.createConnection() ) {

            // Create the queue if needed.
            SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
TIME_OUT_SECONDS);
```

```
            // Create a session with client acknowledge mode.
            Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);

            // Create a producer and consumer.
            MessageProducer producer =
session.createProducer(session.createQueue(QUEUE_NAME));
            MessageConsumer consumer =
session.createConsumer(session.createQueue(QUEUE_NAME));

            // Open the connection.
            connection.start();

            // Send two text messages.
            sendMessage(producer, session, "Message 1");
            sendMessage(producer, session, "Message 2");

            // Receive a message and don't acknowledge it.
            receiveMessage(consumer, false);

            // Receive another message and acknowledge it.
            receiveMessage(consumer, true);

            // Wait for the visibility time out, so that unacknowledged messages
reappear in the queue,
            LOGGER.info("Waiting for visibility timeout...");
            try {
                Thread.sleep(TIME_OUT_MILLIS);
            } catch (InterruptedException e) {
                LOGGER.error("Interrupted while waiting for visibility timeout", e);
                Thread.currentThread().interrupt();
                throw new RuntimeException("Processing interrupted", e);
            }

            /*  We will attempt to receive another message, but none will be
available. This is because in
                CLIENT_ACKNOWLEDGE mode, when we acknowledged the second message,
all previous messages were
                automatically acknowledged as well. Therefore, although we never
directly acknowledged the first
                message, it was implicitly acknowledged when we confirmed the second
one. */
            receiveMessage(consumer, true);
```

```
        } // The connection closes automatically. This also closes the session.
        LOGGER.info("Connection closed.");

    }


    /**
     * Sends a text message using the specified JMS MessageProducer and Session.
     *
     * @param producer    The JMS MessageProducer used to send the message
     * @param session     The JMS Session used to create the text message
     * @param messageText The text content to be sent in the message
     * @throws JMSException If there is an error creating or sending the message
     */
    private static void sendMessage(MessageProducer producer, Session session,
String messageText) throws JMSException {
        // Create a text message and send it.
        producer.send(session.createTextMessage(messageText));
    }

    /**
     * Receives and processes a message from a JMS queue using the specified
consumer.
     * The method waits for a message until the configured timeout period is
reached.
     * If a message is received, it is logged and optionally acknowledged based on
the
     * acknowledge parameter.
     *
     * @param consumer    The JMS MessageConsumer used to receive messages from the
queue
     * @param acknowledge Boolean flag indicating whether to acknowledge the
message.
     *                    If true, the message will be acknowledged after processing
     * @throws JMSException If there is an error receiving, processing, or
acknowledging the message
     */
    private static void receiveMessage(MessageConsumer consumer, boolean
acknowledge) throws JMSException {
        // Receive a message.
        Message message = consumer.receive(TIME_OUT_MILLIS);

        if (message == null) {
            LOGGER.info("Queue is empty!");
```

```
        } else {
            // Since this queue has only text messages, cast the message object and
print the text.
            LOGGER.info("Received: {}    Acknowledged: {}", ((TextMessage)
message).getText(), acknowledge);

            // Acknowledge the message if asked.
            if (acknowledge) message.acknowledge();
        }
    }
```

使用 UNORDERED_ACKNOWLEDGE 模式接收消息。

```
    /**
     * Demonstrates message acknowledgment behavior in UNORDERED_ACKNOWLEDGE mode
with Amazon SQS JMS.
     * In this mode, each message must be explicitly acknowledged regardless of
receive order.
     * Unacknowledged messages return to the queue after the visibility timeout
expires,
     * unlike CLIENT_ACKNOWLEDGE mode where acknowledging one message acknowledges
all previous messages.
     *
     * @throws JMSException         If a JMS-related error occurs during message
operations
     */
    public static void doReceiveMessagesUnorderedAcknowledge() throws JMSException {
        // Create a connection factory.
        SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
                new ProviderConfiguration(),
                SqsClient.create()
        );

        // Create the connection in a try-with-resources statement so that it's
closed automatically.
        try( SQSConnection connection = connectionFactory.createConnection() ) {

            // Create the queue if needed.
            SqsJmsExampleUtils.ensureQueueExists(connection, QUEUE_NAME,
TIME_OUT_SECONDS);

            // Create a session with unordered acknowledge mode.
```

```
            Session session = connection.createSession(false,
SQSSession.UNORDERED_ACKNOWLEDGE);

            // Create the producer and consumer.
            MessageProducer producer =
session.createProducer(session.createQueue(QUEUE_NAME));
            MessageConsumer consumer =
session.createConsumer(session.createQueue(QUEUE_NAME));

            // Open a connection.
            connection.start();

            // Send two text messages.
            sendMessage(producer, session, "Message 1");
            sendMessage(producer, session, "Message 2");

            // Receive a message and don't acknowledge it.
            receiveMessage(consumer, false);

            // Receive another message and acknowledge it.
            receiveMessage(consumer, true);

            // Wait for the visibility time out, so that unacknowledged messages
reappear in the queue.
            LOGGER.info("Waiting for visibility timeout...");
            try {
                Thread.sleep(TIME_OUT_MILLIS);
            } catch (InterruptedException e) {
                LOGGER.error("Interrupted while waiting for visibility timeout", e);
                Thread.currentThread().interrupt();
                throw new RuntimeException("Processing interrupted", e);
            }

            /*  We will attempt to receive another message, and we'll get the first
message again. This occurs
                because in UNORDERED_ACKNOWLEDGE mode, each message requires its own
separate acknowledgment.
                Since we only acknowledged the second message, the first message
remains in the queue for
                redelivery. */
            receiveMessage(consumer, true);

            LOGGER.info("Connection closed.");
        } // The connection closes automatically. This also closes the session.
```

```
    }

    /**
     * Sends a text message to an Amazon SQS queue using JMS.
     *
     * @param producer    The JMS MessageProducer for the queue
     * @param session     The JMS Session for message creation
     * @param messageText The message content
     * @throws JMSException If message creation or sending fails
     */
    private static void sendMessage(MessageProducer producer, Session session,
String messageText) throws JMSException {
        // Create a text message and send it.
        producer.send(session.createTextMessage(messageText));
    }
    /**
     * Synchronously receives a message from an Amazon SQS queue using the JMS API
     * with an acknowledgment parameter.
     *
     * @param consumer    The JMS MessageConsumer for the queue
     * @param acknowledge If true, acknowledges the message after receipt
     * @throws JMSException If message reception or acknowledgment fails
     */
    private static void receiveMessage(MessageConsumer consumer, boolean
acknowledge) throws JMSException {
        // Receive a message.
        Message message = consumer.receive(TIME_OUT_MILLIS);

        if (message == null) {
            LOGGER.info("Queue is empty!");
        } else {
            // Since this queue has only text messages, cast the message object and
print the text.
            LOGGER.info("Received: {}    Acknowledged: {}", ((TextMessage)
message).getText(), acknowledge);

            // Acknowledge the message if asked.
            if (acknowledge) message.acknowledge();
        }
    }
```

使用 Spring 注入依赖关系。

```
package com.example.sqs.jms.spring;

import com.amazon.sqs.javamessaging.SQSConnection;
import com.example.sqs.jms.SqsJmsExampleUtils;
import jakarta.jms.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.NoSuchBeanDefinitionException;
import org.springframework.context.support.FileSystemXmlApplicationContext;

import java.io.File;
import java.net.URL;
import java.util.concurrent.TimeUnit;

/**
 * Demonstrates how to send and receive messages using the Amazon SQS Java Messaging
 Library
 * with Spring Framework integration. This example connects to a standard Amazon SQS
 message
 * queue using Spring's dependency injection to configure the connection and
 messaging components.
 * The application uses the JMS (Java Message Service) API to handle message
 operations.
 */
public class SpringExample {
    private static final Integer POLLING_SECONDS = 15;
    private static final String SPRING_XML_CONFIG_FILE =
 "SpringExampleConfiguration.xml.txt";
    private static final Logger LOGGER =
 LoggerFactory.getLogger(SpringExample.class);

    /**
     * Demonstrates sending and receiving messages through a standard Amazon SQS
 message queue
     * using Spring Framework configuration. This method loads connection settings
 from an XML file,
     * establishes a messaging session using the Amazon SQS Java Messaging Library,
 and processes
     * messages using JMS (Java Message Service) operations. If the queue doesn't
 exist, it will
     * be created automatically.
     *
     * @param args Command line arguments (not used)
```

```
     */
    public static void main(String[] args) {

        URL resource =
SpringExample.class.getClassLoader().getResource(SPRING_XML_CONFIG_FILE);
        File springFile = new File(resource.getFile());
        if (!springFile.exists() || !springFile.canRead()) {
            LOGGER.error("File " + SPRING_XML_CONFIG_FILE + " doesn't exist or isn't
readable.");
            System.exit(1);
        }

        try (FileSystemXmlApplicationContext context =
                    new FileSystemXmlApplicationContext("file://" +
springFile.getAbsolutePath())) {

            Connection connection;
            try {
                connection = context.getBean(Connection.class);
            } catch (NoSuchBeanDefinitionException e) {
                LOGGER.error("Can't find the JMS connection to use: " +
e.getMessage(), e);
                System.exit(2);
                return;
            }

            String queueName;
            try {
                queueName = context.getBean("queueName", String.class);
            } catch (NoSuchBeanDefinitionException e) {
                LOGGER.error("Can't find the name of the queue to use: " +
e.getMessage(), e);
                System.exit(3);
                return;
            }
            try {
                if (connection instanceof SQSConnection) {
                    SqsJmsExampleUtils.ensureQueueExists((SQSConnection) connection,
queueName, SqsJmsExampleUtils.QUEUE_VISIBILITY_TIMEOUT);
                }
                // Create the JMS session.
                Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);
```

```
                SqsJmsExampleUtils.sendTextMessage(session, queueName);
                MessageConsumer consumer =
session.createConsumer(session.createQueue(queueName));

                receiveMessages(consumer);
            } catch (JMSException e) {
                LOGGER.error(e.getMessage(), e);
                throw new RuntimeException(e);
            }
        }   // Spring context autocloses. Managed Spring beans that implement
AutoClosable, such as the
        // 'connection' bean, are also closed.
        LOGGER.info("Context closed");
    }

    /**
     * Continuously checks for and processes messages from a standard Amazon SQS
message queue
     * using the Amazon SQS Java Messaging Library underlying the JMS API. This
method waits for incoming messages,
     * processes them when they arrive, and acknowledges their receipt using JMS
(Java Message
     * Service) operations. The method will stop checking for messages after 15
seconds of
     * inactivity.
     *
     * @param consumer The JMS message consumer used to receive messages from the
queue
     */
    private static void receiveMessages(MessageConsumer consumer) {
        try {
            while (true) {
                LOGGER.info("Waiting for messages...");
                // Wait 15 seconds for a message.
                Message message =
consumer.receive(TimeUnit.SECONDS.toMillis(POLLING_SECONDS));
                if (message == null) {
                    LOGGER.info("Shutting down after {} seconds of silence.",
POLLING_SECONDS);
                    break;
                }
                SqsJmsExampleUtils.handleMessage(message);
                message.acknowledge();
                LOGGER.info("Message acknowledged.");
```

```
            }
        } catch (JMSException e) {
            LOGGER.error("Error receiving from SQS.", e);
        }
    }
}
```

Spring bean 定义。

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
        xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="
http://www.springframework.org/schema/beans http://www.springframework.org/schema/
beans/spring-beans-3.0.xsd
        ">
    <!-- Define the AWS Region -->
    <bean id="region" class="software.amazon.awssdk.regions.Region" factory-
method="of">
        <constructor-arg value="us-east-1"/>
    </bean>

    <bean id="credentialsProviderBean"
 class="software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider"
          factory-method="create"/>

    <bean id="clientBuilder" class="software.amazon.awssdk.services.sqs.SqsClient"
 factory-method="builder"/>

    <bean id="regionSetClientBuilder" factory-bean="clientBuilder" factory-
method="region">
        <constructor-arg ref="region"/>
    </bean>

    <!-- Configure the Builder with Credentials Provider -->
    <bean id="sqsClient" factory-bean="regionSetClientBuilder" factory-
method="credentialsProvider">
        <constructor-arg ref="credentialsProviderBean"/>
    </bean>
```

```
    <bean id="providerConfiguration"
 class="com.amazon.sqs.javamessaging.ProviderConfiguration">
        <property name="numberOfMessagesToPrefetch" value="5"/>
    </bean>

    <bean id="connectionFactory"
 class="com.amazon.sqs.javamessaging.SQSConnectionFactory">
        <constructor-arg ref="providerConfiguration"/>
        <constructor-arg ref="clientBuilder"/>
    </bean>

    <bean id="connection"
          factory-bean="connectionFactory"
          factory-method="createConnection"
          init-method="start"
          destroy-method="close"/>

    <bean id="queueName" class="java.lang.String">
        <constructor-arg value="SQSJMSClientExampleQueue"/>
    </bean>
</beans>
```

一个实用程序类，提供由其它示例使用的常用方法。

```
package com.example.sqs.jms;

import com.amazon.sqs.javamessaging.AmazonSQSMessagingClientWrapper;
import com.amazon.sqs.javamessaging.ProviderConfiguration;
import com.amazon.sqs.javamessaging.SQSConnection;
import com.amazon.sqs.javamessaging.SQSConnectionFactory;
import jakarta.jms.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;

import java.time.Duration;
import java.util.Base64;
import java.util.Map;
```

```
/**
 * This utility class provides helper methods for working with Amazon Simple Queue
 Service (Amazon SQS)
 * through the Java Message Service (JMS) interface. It contains common operations
 for managing message
 * queues and handling message delivery.
 */
public class SqsJmsExampleUtils {
    private static final Logger LOGGER =
 LoggerFactory.getLogger(SqsJmsExampleUtils.class);
    public static final Long QUEUE_VISIBILITY_TIMEOUT = 5L;

    /**
     * This method verifies that a message queue exists and creates it if necessary.
 The method checks for
     * an existing queue first to optimize performance.
     *
     * @param connection The active connection to the messaging service
     * @param queueName The name of the queue to verify or create
     * @param visibilityTimeout The duration in seconds that messages will be hidden
 after being received
     * @throws JMSException If there is an error accessing or creating the queue
     */
    public static void ensureQueueExists(SQSConnection connection, String queueName,
 Long visibilityTimeout) throws JMSException {
        AmazonSQSMessagingClientWrapper client =
 connection.getWrappedAmazonSQSClient();

      /* In most cases, you can do this with just a 'createQueue' call, but
 'getQueueUrl'
      (called by 'queueExists') is a faster operation for the common case where the
 queue
      already exists. Also, many users and roles have permission to call
 'getQueueUrl'
      but don't have permission to call 'createQueue'.
      */
       if( !client.queueExists(queueName) ) {
           CreateQueueRequest createQueueRequest = CreateQueueRequest.builder()
                   .queueName(queueName)
                   .attributes(Map.of(QueueAttributeName.VISIBILITY_TIMEOUT,
 String.valueOf(visibilityTimeout)))
                   .build();
           client.createQueue( createQueueRequest );
```

```
        }
    }

    /**
     * This method sends a simple text message to a specified message queue. It
handles all necessary
     * setup for the message delivery process.
     *
     * @param session The active messaging session used to create and send the
message
     * @param queueName The name of the queue where the message will be sent
     */
    public static void sendTextMessage(Session session, String queueName) {
        // Rest of implementation...

        try {
            MessageProducer producer =
session.createProducer( session.createQueue( queueName) );
            Message message = session.createTextMessage("Hello world!");
            producer.send(message);
        } catch (JMSException e) {
            LOGGER.error( "Error receiving from SQS", e );
        }
    }

    /**
     * This method processes incoming messages and logs their content based on the
message type.
     * It supports text messages, binary data, and Java objects.
     *
     * @param message The message to be processed and logged
     * @throws JMSException If there is an error reading the message content
     */
    public static void handleMessage(Message message) throws JMSException {
        // Rest of implementation...
        LOGGER.info( "Got message {}", message.getJMSMessageID() );
        LOGGER.info( "Content: ");
        if(message instanceof TextMessage txtMessage) {
            LOGGER.info( "\t{}", txtMessage.getText() );
        } else if(message instanceof BytesMessage byteMessage){
            // Assume the length fits in an int - SQS only supports sizes up to 256k
so that
            // should be true
            byte[] bytes = new byte[(int)byteMessage.getBodyLength()];
```

```
            byteMessage.readBytes(bytes);
            LOGGER.info( "\t{}", Base64.getEncoder().encodeToString( bytes ) );
        } else if( message instanceof ObjectMessage) {
            ObjectMessage objMessage = (ObjectMessage) message;
            LOGGER.info( "\t{}", objMessage.getObject() );
        }
    }

    /**
     * This method sets up automatic message processing for a specified queue. It
creates a listener
     * that will receive and handle incoming messages without blocking the main
program.
     *
     * @param session The active messaging session
     * @param queueName The name of the queue to monitor
     * @param connection The active connection to the messaging service
     */
    public static void receiveMessagesAsync(Session session, String queueName,
Connection connection) {
        // Rest of implementation...
        try {
            // Create a consumer for the queue.
            MessageConsumer consumer =
session.createConsumer(session.createQueue(queueName));
            // Provide an implementation of the MessageListener interface, which has
a single 'onMessage' method.
            // We use a lambda expression for the implementation.
            consumer.setMessageListener(message -> {
                try {
                    SqsJmsExampleUtils.handleMessage(message);
                    message.acknowledge();
                } catch (JMSException e) {
                    LOGGER.error("Error processing message: {}", e.getMessage());
                }
            });
            // Start receiving incoming messages.
            connection.start();
        } catch (JMSException e) {
            throw new RuntimeException(e);
        }
        try {
            Thread.sleep(2000);
        } catch (InterruptedException e) {
```

```
            throw new RuntimeException(e);
        }
    }


    /**
     * This method performs cleanup operations after message processing is complete.
It receives
     * any messages in the specified queue, removes the message queue and closes all
     * active connections to prevent resource leaks.
     *
     * @param queueName The name of the queue to be removed
     * @param visibilityTimeout The duration in seconds that messages are hidden
after being received
     * @throws JMSException If there is an error during the cleanup process
     */
    public static void cleanUpExample(String queueName, Long visibilityTimeout)
throws JMSException {
        LOGGER.info("Performing cleanup.");

        SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
                new ProviderConfiguration(),
                SqsClient.create()
        );

        try (SQSConnection connection = connectionFactory.createConnection() ) {
            ensureQueueExists(connection, queueName, visibilityTimeout);
            Session session = connection.createSession(false,
Session.AUTO_ACKNOWLEDGE);

            receiveMessagesAsync(session, queueName, connection);

            SqsClient sqsClient =
connection.getWrappedAmazonSQSClient().getAmazonSQSClient();
            try {
                String queueUrl = sqsClient.getQueueUrl(b ->
b.queueName(queueName)).queueUrl();
                sqsClient.deleteQueue(b -> b.queueUrl(queueUrl));
                LOGGER.info("Queue deleted: {}", queueUrl);
            } catch (SdkException e) {
                LOGGER.error("Error during SQS operations: ", e);
            }
        }
        LOGGER.info("Clean up: Connection closed");
```

```
    }

    /**
     * This method creates a background task that sends multiple messages to a
specified queue
     * after waiting for a set time period. The task operates independently to
ensure efficient
     * message processing without interrupting other operations.
     *
     * @param queueName The name of the queue where messages will be sent
     * @param secondsToWait The number of seconds to wait before sending messages
     * @param numMessages The number of messages to send
     * @param visibilityTimeout The duration in seconds that messages remain hidden
after being received
     * @return A task that can be executed to send the messages
     */
    public static Runnable sendAMessageAsync(String queueName, Long secondsToWait,
Integer numMessages, Long visibilityTimeout) {
        return () -> {
            try {
                Thread.sleep(Duration.ofSeconds(secondsToWait).toMillis());
            } catch (InterruptedException e) {
                Thread.currentThread().interrupt();
                throw new RuntimeException(e);
            }
            try {
                SQSConnectionFactory connectionFactory = new SQSConnectionFactory(
                        new ProviderConfiguration(),
                        SqsClient.create()
                );
                try (SQSConnection connection =
connectionFactory.createConnection()) {
                    ensureQueueExists(connection, queueName, visibilityTimeout);
                    Session session = connection.createSession(false,
Session.CLIENT_ACKNOWLEDGE);
                    for (int i = 1; i <= numMessages; i++) {
                        MessageProducer producer =
session.createProducer(session.createQueue(queueName));
                        producer.send(session.createTextMessage("Hello World " + i +
"!"));
                    }
                }
            } catch (JMSException e) {
                LOGGER.error(e.getMessage(), e);
```

```
                throw new RuntimeException(e);
            }
        };
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - CreateQueue
  - DeleteQueue

处理队列标签

以下代码示例演示如何使用 Amazon SQS 执行标签操作。

适用于 Java 的 SDK 2.x

> **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。

以下示例为队列创建标签、列出标签和移除标签。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.ListQueueTagsResponse;
import software.amazon.awssdk.services.sqs.model.QueueDoesNotExistException;
import software.amazon.awssdk.services.sqs.model.SqsException;

import java.util.Map;
import java.util.UUID;

/**
 * Before running this Java V2 code example, set up your development environment,
 including your credentials. For more
```

```
 * information, see the <a href="https://docs.aws.amazon.com/sdk-for-java/latest/
developer-guide/get-started.html">AWS
 * SDK for Java Developer Guide</a>.
 */
public class TagExamples {
    static final SqsClient sqsClient = SqsClient.create();
    static final String queueName = "TagExamples-queue-" +
 UUID.randomUUID().toString().replace("-", "").substring(0, 20);
    private static final Logger LOGGER = LoggerFactory.getLogger(TagExamples.class);

    public static void main(String[] args) {
        final String queueUrl;
        try {
            queueUrl = sqsClient.createQueue(b ->
 b.queueName(queueName)).queueUrl();
            LOGGER.info("Queue created. The URL is: {}", queueUrl);
        } catch (RuntimeException e) {
            LOGGER.error("Program ending because queue was not created.");
            throw new RuntimeException(e);
        }
        try {
            addTags(queueUrl);
            listTags(queueUrl);
            removeTags(queueUrl);
        } catch (RuntimeException e) {
            LOGGER.error("Program ending because of an error in a method.");
        } finally {
            try {
                sqsClient.deleteQueue(b -> b.queueUrl(queueUrl));
                LOGGER.info("Queue successfully deleted. Program ending.");
                sqsClient.close();
            } catch (RuntimeException e) {
                LOGGER.error("Program ending.");
            } finally {
                sqsClient.close();
            }
        }
    }

    /** This method demonstrates how to use a Java Map to a tag a aueue.
     * @param queueUrl The URL of the queue to tag.
     */
    public static void addTags(String queueUrl) {
        // Build a map of the tags.
```

```
        final Map<String, String> tagsToAdd = Map.of(
                "Team", "Development",
                "Priority", "Beta",
                "Accounting ID", "456def");

        try {
            // Add tags to the queue using a Consumer<TagQueueRequest.Builder>
parameter.
            sqsClient.tagQueue(b -> b
                    .queueUrl(queueUrl)
                    .tags(tagsToAdd)
            );
        } catch (QueueDoesNotExistException e) {
            LOGGER.error("Queue does not exist: {}", e.getMessage(), e);
            throw new RuntimeException(e);
        }
    }

    /** This method demonstrates how to view the tags for a queue.
     * @param queueUrl The URL of the queue whose tags you want to list.
     */
    public static void listTags(String queueUrl) {
        ListQueueTagsResponse response;
        try {
            // Call the listQueueTags method with a
Consumer<ListQueueTagsRequest.Builder> parameter that creates a
ListQueueTagsRequest.
            response = sqsClient.listQueueTags(b -> b
                    .queueUrl(queueUrl));
        } catch (SqsException e) {
            LOGGER.error("Exception thrown: {}", e.getMessage(), e);
            throw new RuntimeException(e);
        }

        // Log the tags.
        response.tags()
                .forEach((k, v) ->
                        LOGGER.info("Key: {} -> Value: {}", k, v));
    }

    /**
     * This method demonstrates how to remove tags from a queue.
     * @param queueUrl The URL of the queue whose tags you want to remove.
     */
```

```
    public static void removeTags(String queueUrl) {
        try {
            // Call the untagQueue method with a Consumer<UntagQueueRequest.Builder>
 parameter.
            sqsClient.untagQueue(b -> b
                    .queueUrl(queueUrl)
                    .tagKeys("Accounting ID") // Remove a single tag.
            );
        } catch (SqsException e) {
            LOGGER.error("Exception thrown: {}", e.getMessage(), e);
            throw new RuntimeException(e);
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - ListQueueTags
  - TagQueue
  - UntagQueue

无服务器示例

通过 Amazon SQS 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 SNS 队列的消息而触发的事件。该函数从事件参数检索消息并记录每条消息的内容。

适用于 Java 的 SDK 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。

通过 Java 将 SQS 事件与 Lambda 结合使用。

```
// 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;
        }

    }
}
```

报告使用 Amazon SQS 触发器进行 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 SQS 队列的事件的 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。

适用于 Java 的 SDK 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 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>();

         for (SQSEvent.SQSMessage message : sqsEvent.getRecords()) {
             try {
                 //process your message
             } catch (Exception e) {
                 //Add failed message identifier to the batchItemFailures list
                 batchItemFailures.add(new
 SQSBatchResponse.BatchItemFailure(message.getMessageId()));
             }
         }
         return new SQSBatchResponse(batchItemFailures);
     }
}
```

使用 SDK for 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

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 基本功能
- 操作
- 场景

开始使用

开始使用 Step Functions

以下代码示例显示了如何开始使用 Step Functions。

适用于 Java 的 SDK 2.x

> **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。

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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ListStateMachines](#)中的。

基本功能

了解基本功能

以下代码示例展示了如何：

- 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。

适用于 Java 的 SDK 2.x

> **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。

```
/**
 * You can obtain the JSON file to create a state machine in the following
 * GitHub location.
 * <p>
 * https://github.com/awsdocs/aws-doc-sdk-examples/tree/main/resources/sample_files
 * <p>
 * To run this code example, place the chat_sfn_state_machine.json file into
 * your project's resources folder.
 * <p>
 * Also, set up your development environment, including your credentials.
 * <p>
 * For information, see this documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 * <p>
 * This Java code example performs the following tasks:
 * <p>
 * 1. Creates an activity.
 * 2. Creates a state machine.
 * 3. Describes the state machine.
 * 4. Starts execution of the state machine and interacts with it.
 * 5. Describes the execution.
 * 6. Delete the activity.
 * 7. Deletes the state machine.
 */
public class StepFunctionsScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
```

```
   public static void main(String[] args) throws Exception {
       final String usage = """

           Usage:
               <roleARN> <activityName> <stateMachineName>

           Where:
               roleName - The name of the IAM role to create for this state
machine.
               activityName - The name of an activity to create.
               stateMachineName - The name of the state machine to create.
               jsonFile - The location of the chat_sfn_state_machine.json file. You
can located it in resources/sample_files.
           """;

       if (args.length != 4) {
           System.out.println(usage);
           System.exit(1);
       }

       String roleName = args[0];
       String activityName = args[1];
       String stateMachineName = args[2];
       String jsonFile = args[3];
       String polJSON = """
           {
               "Version":"2012-10-17",
               "Statement": [
                   {
                       "Sid": "",
                       "Effect": "Allow",
                       "Principal": {
                           "Service": "states.amazonaws.com"
                       },
                       "Action": "sts:AssumeRole"
                   }
               ]
           }
           """;


       Scanner sc = new Scanner(System.in);
       boolean action = false;
```

```
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
            .region(region)
            .build();

        Region regionGl = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
            .region(regionGl)
            .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS Step Functions example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create an activity.");
        String activityArn = createActivity(sfnClient, activityName);
        System.out.println("The ARN of the activity is " + activityArn);
        System.out.println(DASHES);

        // Read the file using FileInputStream
        FileInputStream inputStream = new FileInputStream(jsonFile);
        ObjectMapper mapper = new ObjectMapper();
        JsonNode jsonNode = mapper.readValue(inputStream, JsonNode.class);
        String jsonString = mapper.writeValueAsString(jsonNode);

        // Modify the Resource node.
        ObjectMapper objectMapper = new ObjectMapper();
        JsonNode root = objectMapper.readTree(jsonString);
        ((ObjectNode) root.path("States").path("GetInput")).put("Resource",
activityArn);

        // Convert the modified Java object back to a JSON string.
        String stateDefinition = objectMapper.writeValueAsString(root);
        System.out.println(stateDefinition);

        System.out.println(DASHES);
        System.out.println("2. Create a state machine.");
        String roleARN = createIAMRole(iam, roleName, polJSON);
        String stateMachineArn = createMachine(sfnClient, roleARN, stateMachineName,
stateDefinition);
        System.out.println("The ARN of the state machine is " + stateMachineArn);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("3. Describe the state machine.");
        describeStateMachine(sfnClient, stateMachineArn);
        System.out.println("What should ChatSFN call you?");
        String userName = sc.nextLine();
        System.out.println("Hello " + userName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        // The JSON to pass to the StartExecution call.
        String executionJson = "{ \"name\" : \"" + userName + "\" }";
        System.out.println(executionJson);
        System.out.println("4. Start execution of the state machine and interact
with it.");
        String runArn = startWorkflow(sfnClient, stateMachineArn, executionJson);
        System.out.println("The ARN of the state machine execution is " + runArn);
        List<String> myList;
        while (!action) {
            myList = getActivityTask(sfnClient, activityArn);
            System.out.println("ChatSFN: " + myList.get(1));
            System.out.println(userName + " please specify a value.");
            String myAction = sc.nextLine();
            if (myAction.compareTo("done") == 0)
                action = true;

            System.out.println("You have selected " + myAction);
            String taskJson = "{ \"action\" : \"" + myAction + "\" }";
            System.out.println(taskJson);
            sendTaskSuccess(sfnClient, myList.get(0), taskJson);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Describe the execution.");
        describeExe(sfnClient, runArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Delete the activity.");
        deleteActivity(sfnClient, activityArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Delete the state machines.");
```

```
        deleteMachine(sfnClient, stateMachineArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The AWS Step Functions example scenario is complete.");
        System.out.println(DASHES);
    }

    public static String createIAMRole(IamClient iam, String rolename, String
polJSON) {
        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                .roleName(rolename)
                .assumeRolePolicyDocument(polJSON)
                .description("Created using the AWS SDK for Java")
                .build();

            CreateRoleResponse response = iam.createRole(request);
            return response.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void describeExe(SfnClient sfnClient, String executionArn) {
        try {
            DescribeExecutionRequest executionRequest =
DescribeExecutionRequest.builder()
                .executionArn(executionArn)
                .build();

            String status = "";
            boolean hasSucceeded = false;
            while (!hasSucceeded) {
                DescribeExecutionResponse response =
sfnClient.describeExecution(executionRequest);
                status = response.statusAsString();
                if (status.compareTo("RUNNING") == 0) {
                    System.out.println("The state machine is still running, let's
wait for it to finish.");
                    Thread.sleep(2000);
```

```
                } else if (status.compareTo("SUCCEEDED") == 0) {
                    System.out.println("The Step Function workflow has succeeded");
                    hasSucceeded = true;
                } else {
                    System.out.println("The Status is neither running or
succeeded");
                }
            }
            System.out.println("The Status is " + status);

        } catch (SfnException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void sendTaskSuccess(SfnClient sfnClient, String token, String
json) {
        try {
            SendTaskSuccessRequest successRequest = SendTaskSuccessRequest.builder()
                .taskToken(token)
                .output(json)
                .build();

            sfnClient.sendTaskSuccess(successRequest);

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static List<String> getActivityTask(SfnClient sfnClient, String actArn) {
        List<String> myList = new ArrayList<>();
        GetActivityTaskRequest getActivityTaskRequest =
GetActivityTaskRequest.builder()
            .activityArn(actArn)
            .build();

        GetActivityTaskResponse response =
sfnClient.getActivityTask(getActivityTaskRequest);
        myList.add(response.taskToken());
        myList.add(response.input());
        return myList;
```

```
    }

    public static void deleteActivity(SfnClient sfnClient, String actArn) {
        try {
            DeleteActivityRequest activityRequest = DeleteActivityRequest.builder()
                .activityArn(actArn)
                .build();

            sfnClient.deleteActivity(activityRequest);
            System.out.println("You have deleted " + actArn);

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void describeStateMachine(SfnClient sfnClient, String
stateMachineArn) {
        try {
            DescribeStateMachineRequest stateMachineRequest =
DescribeStateMachineRequest.builder()
                .stateMachineArn(stateMachineArn)
                .build();

            DescribeStateMachineResponse response =
sfnClient.describeStateMachine(stateMachineRequest);
            System.out.println("The name of the State machine is " +
response.name());
            System.out.println("The status of the State machine is " +
response.status());
            System.out.println("The ARN value of the State machine is " +
response.stateMachineArn());
            System.out.println("The role ARN value is " + response.roleArn());

        } catch (SfnException e) {
            System.err.println(e.getMessage());
        }
    }

    public static void deleteMachine(SfnClient sfnClient, String stateMachineArn) {
        try {
            DeleteStateMachineRequest deleteStateMachineRequest =
DeleteStateMachineRequest.builder()
```

```
                .stateMachineArn(stateMachineArn)
                .build();

            sfnClient.deleteStateMachine(deleteStateMachineRequest);
            DescribeStateMachineRequest describeStateMachine =
DescribeStateMachineRequest.builder()
                .stateMachineArn(stateMachineArn)
                .build();

            while (true) {
                DescribeStateMachineResponse response =
sfnClient.describeStateMachine(describeStateMachine);
                System.out.println("The state machine is not deleted yet. The status
is " + response.status());
                Thread.sleep(3000);
            }

        } catch (SfnException | InterruptedException e) {
            System.err.println(e.getMessage());
        }
        System.out.println(stateMachineArn + " was successfully deleted.");
    }

    public static String startWorkflow(SfnClient sfnClient, String stateMachineArn,
String jsonEx) {
        UUID uuid = UUID.randomUUID();
        String uuidValue = uuid.toString();
        try {
            StartExecutionRequest executionRequest = StartExecutionRequest.builder()
                .input(jsonEx)
                .stateMachineArn(stateMachineArn)
                .name(uuidValue)
                .build();

            StartExecutionResponse response =
sfnClient.startExecution(executionRequest);
            return response.executionArn();

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

```
    public static String createMachine(SfnClient sfnClient, String roleARN, String
 stateMachineName, String json) {
        try {
            CreateStateMachineRequest machineRequest =
 CreateStateMachineRequest.builder()
                .definition(json)
                .name(stateMachineName)
                .roleArn(roleARN)
                .type(StateMachineType.STANDARD)
                .build();

            CreateStateMachineResponse response =
 sfnClient.createStateMachine(machineRequest);
            return response.stateMachineArn();

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createActivity(SfnClient sfnClient, String activityName) {
        try {
            CreateActivityRequest activityRequest = CreateActivityRequest.builder()
                .name(activityName)
                .build();

            CreateActivityResponse response =
 sfnClient.createActivity(activityRequest);
            return response.activityArn();

        } catch (SfnException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。

- CreateActivity
- CreateStateMachine
- DeleteActivity
- DeleteStateMachine
- DescribeExecution
- DescribeStateMachine
- GetActivityTask
- ListActivities
- ListStateMachines
- SendTaskSuccess
- StartExecution
- StopExecution

操作

`CreateActivity`

以下代码示例演示了如何使用 `CreateActivity`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateActivity中的。

`CreateStateMachine`

以下代码示例演示了如何使用 `CreateStateMachine`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateStateMachine中的。

`DeleteActivity`

以下代码示例演示了如何使用 `DeleteActivity`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteActivity中的。

`DeleteStateMachine`

以下代码示例演示了如何使用 `DeleteStateMachine`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteStateMachine中的。

`DescribeExecution`

以下代码示例演示了如何使用 `DescribeExecution`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeExecution中的。

`DescribeStateMachine`

以下代码示例演示了如何使用 `DescribeStateMachine`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeStateMachine中的。

GetActivityTask

以下代码示例演示了如何使用 GetActivityTask。

适用于 Java 的 SDK 2.x

> **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。

```
    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;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考GetActivityTask中的。

ListActivities

以下代码示例演示了如何使用 ListActivities。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sfn.SfnClient;
import software.amazon.awssdk.services.sfn.model.ListActivitiesRequest;
import software.amazon.awssdk.services.sfn.model.ListActivitiesResponse;
import software.amazon.awssdk.services.sfn.model.SfnException;
import software.amazon.awssdk.services.sfn.model.ActivityListItem;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class ListActivities {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        SfnClient sfnClient = SfnClient.builder()
                .region(region)
                .build();

        listAllActivites(sfnClient);
        sfnClient.close();
    }

    public static void listAllActivites(SfnClient sfnClient) {
        try {
            ListActivitiesRequest activitiesRequest =
 ListActivitiesRequest.builder()
                    .maxResults(10)
                    .build();

            ListActivitiesResponse response =
 sfnClient.listActivities(activitiesRequest);
            List<ActivityListItem> items = response.activities();
            for (ActivityListItem item : items) {
                System.out.println("The activity ARN is " + item.activityArn());
                System.out.println("The activity name is " + item.name());
            }

        } catch (SfnException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListActivities中的。

`ListExecutions`

以下代码示例演示了如何使用 `ListExecutions`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListExecutions中的。

`ListStateMachines`

以下代码示例演示了如何使用 `ListStateMachines`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListStateMachines中的。

SendTaskSuccess

以下代码示例演示了如何使用 SendTaskSuccess。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SendTaskSuccess中的。

`StartExecution`

以下代码示例演示了如何使用 `StartExecution`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartExecution中的。

场景

使用 Step Functions 调用 Lambda 函数

以下代码示例说明如何创建按顺序调用 AWS Lambda 函数的 AWS Step Functions 状态机。

适用于 Java 的 SDK 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
- Amazon SES
- Step Functions

AWS STS 使用适用于 Java 的 SDK 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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从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`

以下代码示例演示了如何使用 `AssumeRole`。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.services.sts.model.AssumeRoleRequest;
import software.amazon.awssdk.services.sts.model.StsException;
import software.amazon.awssdk.services.sts.model.AssumeRoleResponse;
import software.amazon.awssdk.services.sts.model.Credentials;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * To make this code example work, create a Role that you want to assume.
 * Then define a Trust Relationship in the AWS Console. You can use this as an
 * example:
 *
 * {
 * "Version":"2012-10-17",
 * "Statement": [
 * {
 * "Effect": "Allow",
 * "Principal": {
 * "AWS": "<Specify the ARN of your IAM user you are using in this code example>"
 * },
 * "Action": "sts:AssumeRole"
 * }
 * ]
```

```
 * }
 *
 * For more information, see "Editing the Trust Relationship for an Existing
 * Role" in the AWS Directory Service guide.
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class AssumeRole {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <roleArn> <roleSessionName>\s

                Where:
                    roleArn - The Amazon Resource Name (ARN) of the role to assume
 (for example, arn:aws:iam::000008047983:role/s3role).\s
                    roleSessionName - An identifier for the assumed role session
 (for example, mysession).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleArn = args[0];
        String roleSessionName = args[1];
        Region region = Region.US_EAST_1;
        StsClient stsClient = StsClient.builder()
                .region(region)
                .build();

        assumeGivenRole(stsClient, roleArn, roleSessionName);
        stsClient.close();
    }

    public static void assumeGivenRole(StsClient stsClient, String roleArn, String
 roleSessionName) {
        try {
```

```
            AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                    .roleArn(roleArn)
                    .roleSessionName(roleSessionName)
                    .build();

            AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
            Credentials myCreds = roleResponse.credentials();

            // Display the time when the temp creds expire.
            Instant exTime = myCreds.expiration();
            String tokenInfo = myCreds.sessionToken();

            // Convert the Instant to readable date.
            DateTimeFormatter formatter =
 DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                    .withLocale(Locale.US)
                    .withZone(ZoneId.systemDefault());

            formatter.format(exTime);
            System.out.println("The token " + tokenInfo + "  expires on " + exTime);

        } catch (StsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考AssumeRole中的。

支持 使用适用于 Java 的 SDK 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 支持。 AWS SDK for Java 2.x

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 基本功能
- 操作

开始使用

你好 支持

以下代码示例展示了如何开始使用 支持。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeServices中的。

基本功能

了解基本功能

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。

适用于 Java 的 SDK 2.x

> **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。

运行各种 支持 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 = "C:\\AWS\\test.txt" ; //args[0];
        Region region = Region.US_WEST_2;
        SupportClient supportClient = SupportClient.builder()
                .region(region)
```

```
                .build();

        System.out.println(DASHES);
        System.out.println("***** Welcome to the AWS Support case example
scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Get and display available services.");
        List<String> sevCatList = displayServices(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Get and display Support severity levels.");
        String sevLevel = displaySevLevels(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create a support case using the selected service,
category, and severity level.");
        String caseId = createSupportCase(supportClient, sevCatList, sevLevel);
        if (caseId.compareTo("") == 0) {
            System.out.println("A support case was not successfully created!");
            System.exit(1);
        } else
            System.out.println("Support case " + caseId + " was successfully
created!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get open support cases.");
        getOpenCase(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Create an attachment set with a generated file to add
to the case.");
        String attachmentSetId = addAttachment(supportClient, fileAttachment);
        System.out.println("The Attachment Set id value is" + attachmentSetId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Add communication with the attachment to the support
case.");
```

```
        addAttachSupportCase(supportClient, caseId, attachmentSetId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. List the communications of the support case.");
        String attachId = listCommunications(supportClient, caseId);
        System.out.println("The Attachment id value is" + attachId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Describe the attachment set included with the
communication.");
        describeAttachment(supportClient, attachId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Resolve the support case.");
        resolveSupportCase(supportClient, caseId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get a list of resolved cases for the current day.");
        getResolvedCase(supportClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("***** This Scenario has successfully completed");
        System.out.println(DASHES);
    }

    public static void getResolvedCase(SupportClient supportClient) {
        try {
            // Specify the start and end time.
            Instant now = Instant.now();
            java.time.LocalDate.now();
            Instant yesterday = now.minus(1, ChronoUnit.DAYS);

            DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
                    .maxResults(30)
                    .afterTime(yesterday.toString())
                    .beforeTime(now.toString())
                    .includeResolvedCases(true)
                    .build();
```

```
            DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
            List<CaseDetails> cases = response.cases();
            for (CaseDetails sinCase : cases) {
                if (sinCase.status().compareTo("resolved") == 0)
                    System.out.println("The case status is " + sinCase.status());
            }

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void resolveSupportCase(SupportClient supportClient, String
caseId) {
        try {
            ResolveCaseRequest caseRequest = ResolveCaseRequest.builder()
                    .caseId(caseId)
                    .build();

            ResolveCaseResponse response = supportClient.resolveCase(caseRequest);
            System.out.println("The status of case " + caseId + " is " +
response.finalCaseStatus());

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static void describeAttachment(SupportClient supportClient, String
attachId) {
        try {
            DescribeAttachmentRequest attachmentRequest =
DescribeAttachmentRequest.builder()
                    .attachmentId(attachId)
                    .build();

            DescribeAttachmentResponse response =
supportClient.describeAttachment(attachmentRequest);
            System.out.println("The name of the file is " +
response.attachment().fileName());
```

```
        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static String listCommunications(SupportClient supportClient, String
caseId) {
        try {
            String attachId = null;
            DescribeCommunicationsRequest communicationsRequest =
DescribeCommunicationsRequest.builder()
                    .caseId(caseId)
                    .maxResults(10)
                    .build();

            DescribeCommunicationsResponse response =
supportClient.describeCommunications(communicationsRequest);
            List<Communication> communications = response.communications();
            for (Communication comm : communications) {
                System.out.println("the body is: " + comm.body());

                // Get the attachment id value.
                List<AttachmentDetails> attachments = comm.attachmentSet();
                for (AttachmentDetails detail : attachments) {
                    attachId = detail.attachmentId();
                }
            }
            return attachId;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    public static void addAttachSupportCase(SupportClient supportClient, String
caseId, String attachmentSetId) {
        try {
            AddCommunicationToCaseRequest caseRequest =
AddCommunicationToCaseRequest.builder()
                    .caseId(caseId)
```

```
                    .attachmentSetId(attachmentSetId)
                    .communicationBody("Please refer to attachment for details.")
                    .build();

            AddCommunicationToCaseResponse response =
supportClient.addCommunicationToCase(caseRequest);
            if (response.result())
                System.out.println("You have successfully added a communication to
an AWS Support case");
            else
                System.out.println("There was an error adding the communication to
an AWS Support case");

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static String addAttachment(SupportClient supportClient, String
fileAttachment) {
        try {
            File myFile = new File(fileAttachment);
            InputStream sourceStream = new FileInputStream(myFile);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            Attachment attachment = Attachment.builder()
                    .fileName(myFile.getName())
                    .data(sourceBytes)
                    .build();

            AddAttachmentsToSetRequest setRequest =
AddAttachmentsToSetRequest.builder()
                    .attachments(attachment)
                    .build();

            AddAttachmentsToSetResponse response =
supportClient.addAttachmentsToSet(setRequest);
            return response.attachmentSetId();

        } catch (SupportException | FileNotFoundException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
```

```
        return "";
    }

    public static void getOpenCase(SupportClient supportClient) {
        try {
            // Specify the start and end time.
            Instant now = Instant.now();
            java.time.LocalDate.now();
            Instant yesterday = now.minus(1, ChronoUnit.DAYS);

            DescribeCasesRequest describeCasesRequest =
DescribeCasesRequest.builder()
                    .maxResults(20)
                    .afterTime(yesterday.toString())
                    .beforeTime(now.toString())
                    .build();

            DescribeCasesResponse response =
supportClient.describeCases(describeCasesRequest);
            List<CaseDetails> cases = response.cases();
            for (CaseDetails sinCase : cases) {
                System.out.println("The case status is " + sinCase.status());
                System.out.println("The case Id is " + sinCase.caseId());
                System.out.println("The case subject is " + sinCase.subject());
            }

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
    }

    public static String createSupportCase(SupportClient supportClient, List<String>
sevCatList, String sevLevel) {
        try {
            String serviceCode = sevCatList.get(0);
            String caseCat = sevCatList.get(1);
            CreateCaseRequest caseRequest = CreateCaseRequest.builder()
                    .categoryCode(caseCat.toLowerCase())
                    .serviceCode(serviceCode.toLowerCase())
                    .severityCode(sevLevel.toLowerCase())
                    .communicationBody("Test issue with " +
serviceCode.toLowerCase())
                    .subject("Test case, please ignore")
```

```
                    .language("en")
                    .issueType("technical")
                    .build();

            CreateCaseResponse response = supportClient.createCase(caseRequest);
            return response.caseId();

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    public static String displaySevLevels(SupportClient supportClient) {
        try {
            DescribeSeverityLevelsRequest severityLevelsRequest =
DescribeSeverityLevelsRequest.builder()
                    .language("en")
                    .build();

            DescribeSeverityLevelsResponse response =
supportClient.describeSeverityLevels(severityLevelsRequest);
            List<SeverityLevel> severityLevels = response.severityLevels();
            String levelName = null;
            for (SeverityLevel sevLevel : severityLevels) {
                System.out.println("The severity level name is: " +
sevLevel.name());
                if (sevLevel.name().compareTo("High") == 0)
                    levelName = sevLevel.name();
            }
            return levelName;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return "";
    }

    // Return a List that contains a Service name and Category name.
    public static List<String> displayServices(SupportClient supportClient) {
        try {
```

```
            DescribeServicesRequest servicesRequest =
 DescribeServicesRequest.builder()
                    .language("en")
                    .build();

            DescribeServicesResponse response =
 supportClient.describeServices(servicesRequest);
            String serviceCode = null;
            String catName = null;
            List<String> sevCatList = new ArrayList<>();
            List<Service> services = response.services();

            System.out.println("Get the first 10 services");
            int index = 1;
            for (Service service : services) {
                if (index == 11)
                    break;

                System.out.println("The Service name is: " + service.name());
                if (service.name().compareTo("Account") == 0)
                    serviceCode = service.code();

                // Get the Categories for this service.
                List<Category> categories = service.categories();
                for (Category cat : categories) {
                    System.out.println("The category name is: " + cat.name());
                    if (cat.name().compareTo("Security") == 0)
                        catName = cat.name();
                }
                index++;
            }

            // Push the two values to the list.
            sevCatList.add(serviceCode);
            sevCatList.add(catName);
            return sevCatList;

        } catch (SupportException e) {
            System.out.println(e.getLocalizedMessage());
            System.exit(1);
        }
        return null;
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - AddAttachmentsToSet
  - AddCommunicationToCase
  - CreateCase
  - DescribeAttachment
  - DescribeCases
  - DescribeCommunications
  - DescribeServices
  - DescribeSeverityLevels
  - ResolveCase

操作

`AddAttachmentsToSet`

以下代码示例演示了如何使用 `AddAttachmentsToSet`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考AddAttachmentsToSet中的。

`AddCommunicationToCase`

以下代码示例演示了如何使用 `AddCommunicationToCase`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[AddCommunicationToCase](#)中的。

`CreateCase`

以下代码示例演示了如何使用 `CreateCase`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateCase中的。

`DescribeAttachment`

以下代码示例演示了如何使用 `DescribeAttachment`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeAttachment中的。

`DescribeCases`

以下代码示例演示了如何使用 `DescribeCases`。

适用于 Java 的 SDK 2.x

> **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。

```
    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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeCases中的。

`DescribeCommunications`

以下代码示例演示了如何使用 `DescribeCommunications`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeCommunications中的。

`DescribeServices`

以下代码示例演示了如何使用 `DescribeServices`。

适用于 Java 的 SDK 2.x

> **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。

```
    // 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeServices中的。

DescribeSeverityLevels

以下代码示例演示了如何使用 DescribeSeverityLevels。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeSeverityLevels中的。

`ResolveCase`

以下代码示例演示了如何使用 `ResolveCase`。

适用于 Java 的 SDK 2.x

> **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。

```
   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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ResolveCase中的。

使用 SDK for 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。

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 开始使用
- 基本功能
- 操作

开始使用

开始使用 Systems Manager

以下代码示例展示了如何开始使用 Systems Manager。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ListDocuments中的。

基本功能

了解基本功能

以下代码示例展示了如何：

- 创建维护时段。
- 修改维护时段计划。
- 创建文档。
- 向指定 EC2 实例发送命令。
- 创建一个 OpsItem.
- 更新并解决 OpsItem。
- 删除维护时段 OpsItem、和文档。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.services.ssm.model.DocumentAlreadyExistsException;
import software.amazon.awssdk.services.ssm.model.SsmException;

import java.util.Map;
import java.util.Scanner;
public class SSMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0", "-");
    private static final String ROLES_STACK = "SsmStack3`1";

    public static void main(String[] args) {
        String usage = """
            Usage:
              <title> <source> <category> <severity>

            Where:
                title - The title of the parameter (default is Disk Space Alert).
                source - The source of the parameter (default is EC2).
```

```
                category - The category of the parameter. Valid values are
'Availability', 'Cost', 'Performance', 'Recovery', 'Security' (default is
Performance).
                severity - The severity of the parameter. Severity should be a
number from 1 to 4 (default is 2).
        """;

        Scanner scanner = new Scanner(System.in);
        SSMActions actions = new SSMActions();
        String documentName;
        String windowName;

        System.out.println("Use AWS CloudFormation to create the EC2 instance that
is required for this scenario.");
        CloudFormationHelper.deployCloudFormationStack(ROLES_STACK);
        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputsAsync(ROLES_STACK).join();
        String instanceId = stackOutputs.get("InstanceId");
        System.out.println("The Instance ID: " + instanceId +" was created.");
        String title = "Disk Space Alert" ;
        String source = "EC2" ;
        String category = "Availability" ;
        String severity = "2" ;

        System.out.println(DASHES);
        System.out.println("""
                Welcome to the AWS Systems Manager SDK Basics scenario.
                This Java program demonstrates how to interact with AWS Systems
Manager using the AWS SDK for Java (v2).
                AWS Systems Manager is the operations hub for your AWS applications
and resources and a secure end-to-end management solution.
                The program's primary functionalities include creating a maintenance
window, creating a document, sending a command to a document,
                listing documents, listing commands, creating an OpsItem, modifying
an OpsItem, and deleting AWS SSM resources.
                Upon completion of the program, all AWS resources are cleaned up.
                Let's get started...

                """);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("1. Create an SSM maintenance window.");
```

```
        System.out.println("Please enter the maintenance window name (default is
ssm-maintenance-window):");
        String win = scanner.nextLine();
        windowName = win.isEmpty() ? "ssm-maintenance-window" : win;
        String winId = null;
        try {
            winId = actions.createMaintenanceWindow(windowName);
            waitForInputToContinue(scanner);
            System.out.println("The maintenance window ID is: " + winId);
        } catch (DocumentAlreadyExistsException e) {
            System.err.println("The SSM maintenance window already exists.
Retrieving existing window ID...");
            String existingWinId = actions.createMaintenanceWindow(windowName);
            System.out.println("Existing window ID: " + existingWinId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("2. Modify the maintenance window by changing the
schedule");
        waitForInputToContinue(scanner);
        try {
            actions.updateSSMMaintenanceWindow(winId, windowName);
            waitForInputToContinue(scanner);
            System.out.println("The SSM maintenance window was successfully
updated");
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("3. Create an SSM document that defines the actions that
Systems Manager performs on your managed nodes.");
```

```
        System.out.println("Please enter the document name (default is
ssmdocument):");
        String doc = scanner.nextLine();
        documentName = doc.isEmpty() ? "ssmdocument" : doc;
        try {
            actions.createSSMDoc(documentName);
            waitForInputToContinue(scanner);
            System.out.println("The SSM document was successfully created");
        } catch (DocumentAlreadyExistsException e) {
            System.err.println("The SSM document already exists. Moving on");
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("4. Now we are going to run a command on an EC2
instance");
        waitForInputToContinue(scanner);
        String commandId="";
        try {
            commandId = actions.sendSSMCommand(documentName, instanceId);
            waitForInputToContinue(scanner);
            System.out.println("The command was successfully sent. Command ID: " +
commandId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
        } catch (InterruptedException e) {
            System.err.println("Thread was interrupted: " + e.getMessage());
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println("5. Lets get the time when the specific command was sent
to the specific managed node");
        waitForInputToContinue(scanner);
        try {
            actions.displayCommands(commandId);
```

```
            System.out.println("The command invocations were successfully
displayed.");
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
             6. Now we will create an SSM OpsItem.
             A SSM OpsItem is a feature provided by Amazon's Systems Manager (SSM)
service.
             It is a type of operational data item that allows you to manage and
track various operational issues,
             events, or tasks within your AWS environment.

             You can create OpsItems to track and manage operational issues as they
arise.
             For example, you could create an OpsItem whenever your application
detects a critical error
             or an anomaly in your infrastructure.
            """);

        waitForInputToContinue(scanner);
        String opsItemId;
        try {
            opsItemId = actions.createSSMOpsItem(title, source, category, severity);
            System.out.println(opsItemId + " was created");
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("7. Now we will update the SSM OpsItem "+opsItemId);
        waitForInputToContinue(scanner);
        String description = "An update to "+opsItemId ;
        try {
            actions.updateOpsItem(opsItemId, title, description);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }

        System.out.println(DASHES);
        System.out.println("8. Now we will get the status of the SSM OpsItem
"+opsItemId);
        waitForInputToContinue(scanner);
        try {
            actions.describeOpsItems(opsItemId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }

        System.out.println(DASHES);
        System.out.println("9. Now we will resolve the SSM OpsItem "+opsItemId);
        waitForInputToContinue(scanner);
        try {
            actions.resolveOpsItem(opsItemId);
        } catch (SsmException e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }

        System.out.println(DASHES);
        System.out.println("10. Would you like to delete the AWS Systems Manager
resources? (y/n)");
        String delAns = scanner.nextLine().trim();
```

```
        if (delAns.equalsIgnoreCase("y")) {
            System.out.println("You selected to delete the resources.");
            waitForInputToContinue(scanner);
            try {
                actions.deleteMaintenanceWindow(winId);
                actions.deleteDoc(documentName);
            } catch (SsmException e) {
                System.err.println("SSM error: " + e.getMessage());
                return;
            } catch (RuntimeException e) {
                System.err.println("Unexpected error: " + e.getMessage());
                return;
            }
        } else {
            System.out.println("The AWS Systems Manager resources will not be
 deleted");
        }
        System.out.println(DASHES);
        CloudFormationHelper.destroyCloudFormationStack(ROLES_STACK);
        System.out.println("This concludes the AWS Systems Manager SDK Basics
 scenario.");
        System.out.println(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println("");
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                System.out.println("Continuing with the program...");
                System.out.println("");
                break;
            } else {
                // Handle invalid input.
                System.out.println("Invalid input. Please try again.");
            }
        }
    }
}
```

Systems Manager SDK 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_WEST_2)
                    .httpClient(httpClient)
                    .overrideConfiguration(overrideConfig)
                    .build();
        }
        return ssmAsyncClient;
    }

    /**
     * Deletes an AWS SSM document asynchronously.
     *
     * @param documentName The name of the document to delete.
     * <p>
     * This method initiates an asynchronous request to delete an SSM document.
     * If an exception occurs, it handles the error appropriately.
     */
    public void deleteDoc(String documentName) {
        DeleteDocumentRequest documentRequest = DeleteDocumentRequest.builder()
                .name(documentName)
```

```
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().deleteDocument(documentRequest)
                    .thenAccept(response -> {
                        System.out.println("The SSM document was successfully
deleted.");
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }

    /**
     * Deletes an AWS SSM Maintenance Window asynchronously.
     *
     * @param winId The ID of the Maintenance Window to delete.
     * <p>
     * This method initiates an asynchronous request to delete an SSM Maintenance
Window.
     * If an exception occurs, it handles the error appropriately.
     */
    public void deleteMaintenanceWindow(String winId) {
        DeleteMaintenanceWindowRequest windowRequest =
DeleteMaintenanceWindowRequest.builder()
                .windowId(winId)
```

```
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().deleteMaintenanceWindow(windowRequest)
                    .thenAccept(response -> {
                        System.out.println("The maintenance window was successfully
deleted.");
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }

    /**
     * Resolves an AWS SSM OpsItem asynchronously.
     *
     * @param opsID The ID of the OpsItem to resolve.
     * <p>
     * This method initiates an asynchronous request to resolve an SSM OpsItem.
     * If an exception occurs, it handles the error appropriately.
     */
    public void resolveOpsItem(String opsID) {
        UpdateOpsItemRequest opsItemRequest = UpdateOpsItemRequest.builder()
                .opsItemId(opsID)
                .status(OpsItemStatus.RESOLVED)
                .build();
```

```
        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().updateOpsItem(opsItemRequest)
                    .thenAccept(response -> {
                        System.out.println("OpsItem resolved successfully.");
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }

    /**
     * Describes AWS SSM OpsItems asynchronously.
     *
     * @param key The key to filter OpsItems by (e.g., OPS_ITEM_ID).
     *
     * This method initiates an asynchronous request to describe SSM OpsItems.
     * If the request is successful, it prints the title and status of each OpsItem.
     * If an exception occurs, it handles the error appropriately.
     */
    public void describeOpsItems(String key) {
        OpsItemFilter filter = OpsItemFilter.builder()
                .key(OpsItemFilterKey.OPS_ITEM_ID)
                .values(key)
                .operator(OpsItemFilterOperator.EQUAL)
                .build();
```

```
        DescribeOpsItemsRequest itemsRequest = DescribeOpsItemsRequest.builder()
                .maxResults(10)
                .opsItemFilters(filter)
                .build();

        CompletableFuture<Void> future = CompletableFuture.runAsync(() -> {
            getAsyncClient().describeOpsItems(itemsRequest)
                    .thenAccept(itemsResponse -> {
                        List<OpsItemSummary> items =
itemsResponse.opsItemSummaries();
                        for (OpsItemSummary item : items) {
                            System.out.println("The item title is " + item.title() +
" and the status is " + item.status().toString());
                        }
                    })
                    .exceptionally(ex -> {
                        throw new CompletionException(ex);
                    }).join();
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }

    /**
     * Updates the AWS SSM OpsItem asynchronously.
     *
     * @param opsItemId The ID of the OpsItem to update.
     * @param title The new title of the OpsItem.
```

```
     * @param description The new description of the OpsItem.
     * <p>
     * This method initiates an asynchronous request to update an SSM OpsItem.
     * If the request is successful, it completes without returning a value.
     * If an exception occurs, it handles the error appropriately.
     */
    public void updateOpsItem(String opsItemId, String title, String description) {
        Map<String, OpsItemDataValue> operationalData = new HashMap<>();
        operationalData.put("key1",
 OpsItemDataValue.builder().value("value1").build());
        operationalData.put("key2",
 OpsItemDataValue.builder().value("value2").build());

        CompletableFuture<Void> future = getOpsItem(opsItemId).thenCompose(opsItem -
> {
            UpdateOpsItemRequest request = UpdateOpsItemRequest.builder()
                    .opsItemId(opsItemId)
                    .title(title)
                    .operationalData(operationalData)
                    .status(opsItem.statusAsString())
                    .description(description)
                    .build();

            return getAsyncClient().updateOpsItem(request).thenAccept(response -> {
                System.out.println(opsItemId + " updated successfully.");
            }).exceptionally(ex -> {
                throw new CompletionException(ex);
            });
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
 ex;
            if (cause instanceof SsmException) {
                throw new RuntimeException("SSM error: " + cause.getMessage(),
 cause);
            } else {
                throw new RuntimeException("Unexpected error: " +
 cause.getMessage(), cause);
            }
        });

        try {
            future.join();
        } catch (CompletionException ex) {
```

```
            throw ex.getCause() instanceof RuntimeException ? (RuntimeException)
ex.getCause() : ex;
        }
    }


    private static CompletableFuture<OpsItem> getOpsItem(String opsItemId) {
        GetOpsItemRequest request =
GetOpsItemRequest.builder().opsItemId(opsItemId).build();
        return
getAsyncClient().getOpsItem(request).thenApply(GetOpsItemResponse::opsItem);
    }

    /**
     * Creates an SSM OpsItem asynchronously.
     *
     * @param title The title of the OpsItem.
     * @param source The source of the OpsItem.
     * @param category The category of the OpsItem.
     * @param severity The severity of the OpsItem.
     * @return The ID of the created OpsItem.
     * <p>
     * This method initiates an asynchronous request to create an SSM OpsItem.
     * If the request is successful, it returns the OpsItem ID.
     * If an exception occurs, it handles the error appropriately.
     */
    public String createSSMOpsItem(String title, String source, String category,
String severity) {
        CreateOpsItemRequest opsItemRequest = CreateOpsItemRequest.builder()
                .description("Created by the SSM Java API")
                .title(title)
                .source(source)
                .category(category)
                .severity(severity)
                .build();

        CompletableFuture<CreateOpsItemResponse> future =
getAsyncClient().createOpsItem(opsItemRequest);

        try {
            CreateOpsItemResponse response = future.join();
            return response.opsItemId();
        } catch (CompletionException e) {
            Throwable cause = e.getCause();
```

```
            if (cause instanceof SsmException) {
                throw (SsmException) cause;
            } else {
                throw new RuntimeException(cause);
            }
        }
    }

    /**
     * Displays the date and time when the specific command was invoked.
     *
     * @param commandId The ID of the command to describe.
     * <p>
     * This method initiates an asynchronous request to list command invocations and
prints the date and time of each command invocation.
     * If an exception occurs, it handles the error appropriately.
     */
    public void displayCommands(String commandId) {
        ListCommandInvocationsRequest commandInvocationsRequest =
ListCommandInvocationsRequest.builder()
                .commandId(commandId)
                .build();

        CompletableFuture<ListCommandInvocationsResponse> future =
getAsyncClient().listCommandInvocations(commandInvocationsRequest);
        future.thenAccept(response -> {
            List<CommandInvocation> commandList = response.commandInvocations();
            DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss").withZone(ZoneId.systemDefault());
            for (CommandInvocation invocation : commandList) {
                System.out.println("The time of the command invocation is " +
formatter.format(invocation.requestedDateTime()));
            }
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof SsmException) {
                throw (SsmException) cause;
            } else {
                throw new RuntimeException(cause);
            }
        }).join();
    }
```

```
   /**
    * Sends a SSM command to a managed node asynchronously.
    *
    * @param documentName The name of the document to use.
    * @param instanceId The ID of the instance to send the command to.
    * @return The command ID.
    * <p>
    * This method initiates asynchronous requests to send a SSM command to a
managed node.
    * It waits until the document is active, sends the command, and checks the
command execution status.
    */
   public String sendSSMCommand(String documentName, String instanceId) throws
InterruptedException, SsmException {
       // Before we use Document to send a command - make sure it is active.
       CompletableFuture<Void> documentActiveFuture = CompletableFuture.runAsync(()
-> {
           boolean isDocumentActive = false;
           DescribeDocumentRequest request = DescribeDocumentRequest.builder()
                   .name(documentName)
                   .build();

           while (!isDocumentActive) {
               CompletableFuture<DescribeDocumentResponse> response =
getAsyncClient().describeDocument(request);
               String documentStatus = response.join().document().statusAsString();
               if (documentStatus.equals("Active")) {
                   System.out.println("The SSM document is active and ready to
use.");
                   isDocumentActive = true;
               } else {
                   System.out.println("The SSM document is not active. Status: " +
documentStatus);
                   try {
                       Thread.sleep(5000);
                   } catch (InterruptedException e) {
                       throw new RuntimeException(e);
                   }
               }
           }
       });

       documentActiveFuture.join();
```

```
        // Create the SendCommandRequest.
        SendCommandRequest commandRequest = SendCommandRequest.builder()
                .documentName(documentName)
                .instanceIds(instanceId)
                .build();

        // Send the command.
        CompletableFuture<SendCommandResponse> commandFuture =
getAsyncClient().sendCommand(commandRequest);
        final String[] commandId = {null};

        commandFuture.whenComplete((commandResponse, ex) -> {
            if (commandResponse != null) {
                commandId[0] = commandResponse.command().commandId();
                System.out.println("Command ID: " + commandId[0]);

                // Wait for the command execution to complete.
                GetCommandInvocationRequest invocationRequest =
GetCommandInvocationRequest.builder()
                        .commandId(commandId[0])
                        .instanceId(instanceId)
                        .build();

                try {
                    System.out.println("Wait 5 secs");
                    TimeUnit.SECONDS.sleep(5);

                    // Retrieve the command execution details.
                    CompletableFuture<GetCommandInvocationResponse> invocationFuture
= getAsyncClient().getCommandInvocation(invocationRequest);
                    invocationFuture.whenComplete((commandInvocationResponse,
invocationEx) -> {
                        if (commandInvocationResponse != null) {
                            // Check the status of the command execution.
                            CommandInvocationStatus status =
commandInvocationResponse.status();
                            if (status == CommandInvocationStatus.SUCCESS) {
                                System.out.println("Command execution successful");
                            } else {
                                System.out.println("Command execution failed.
Status: " + status);
                            }
                        } else {
```

```
                            Throwable invocationCause = (invocationEx instanceof
CompletionException) ? invocationEx.getCause() : invocationEx;
                            throw new CompletionException(invocationCause);
                        }
                    }).join();
                } catch (InterruptedException e) {
                    throw new RuntimeException(e);
                }
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof SsmException) {
                    throw (SsmException) cause;
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();

        return commandId[0];
    }

    /**
     * Creates an AWS SSM document asynchronously.
     *
     * @param docName The name of the document to create.
     * <p>
     * This method initiates an asynchronous request to create an SSM document.
     * If the request is successful, it prints the document status.
     * If an exception occurs, it handles the error appropriately.
     */
    public void createSSMDoc(String docName) throws SsmException {
        String jsonData = """
        {
        "schemaVersion": "2.2",
        "description": "Run a simple shell command",
        "mainSteps": [
            {
                "action": "aws:runShellScript",
                "name": "runEchoCommand",
                "inputs": {
                  "runCommand": [
                    "echo 'Hello, world!'"
                  ]
```

```
                }
              }
            ]
        }
        """;

        CreateDocumentRequest request = CreateDocumentRequest.builder()
                .content(jsonData)
                .name(docName)
                .documentType(DocumentType.COMMAND)
                .build();

        CompletableFuture<CreateDocumentResponse> future =
getAsyncClient().createDocument(request);
        future.thenAccept(response -> {
            System.out.println("The status of the SSM document is " +
response.documentDescription().status());
        }).exceptionally(ex -> {
            Throwable cause = (ex instanceof CompletionException) ? ex.getCause() :
ex;
            if (cause instanceof DocumentAlreadyExistsException) {
                throw new CompletionException(cause);
            } else if (cause instanceof SsmException) {
                throw new CompletionException(cause);
            } else {
                throw new RuntimeException(cause);
            }
        }).join();
    }

    /**
     * Updates an SSM maintenance window asynchronously.
     *
     * @param id The ID of the maintenance window to update.
     * @param name The new name for the maintenance window.
     * <p>
     * This method initiates an asynchronous request to update an SSM maintenance
window.
     * If the request is successful, it prints a success message.
     * If an exception occurs, it handles the error appropriately.
     */
    public void updateSSMMaintenanceWindow(String id, String name) throws
SsmException {
```

```
        UpdateMaintenanceWindowRequest updateRequest =
UpdateMaintenanceWindowRequest.builder()
                .windowId(id)
                .allowUnassociatedTargets(true)
                .duration(24)
                .enabled(true)
                .name(name)
                .schedule("cron(0 0 ? * MON *)")
                .build();

        CompletableFuture<UpdateMaintenanceWindowResponse> future =
getAsyncClient().updateMaintenanceWindow(updateRequest);
        future.whenComplete((response, ex) -> {
            if (response != null) {
                System.out.println("The SSM maintenance window was successfully
updated");
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof SsmException) {
                    throw new CompletionException(cause);
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();
    }

    /**
     * Creates an SSM maintenance window asynchronously.
     *
     * @param winName The name of the maintenance window.
     * @return The ID of the created or existing maintenance window.
     * <p>
     * This method initiates an asynchronous request to create an SSM maintenance
window.
     * If the request is successful, it prints the maintenance window ID.
     * If an exception occurs, it handles the error appropriately.
     */
    public String createMaintenanceWindow(String winName) throws SsmException,
DocumentAlreadyExistsException {
        CreateMaintenanceWindowRequest request =
CreateMaintenanceWindowRequest.builder()
                .name(winName)
```

```
                .description("This is my maintenance window")
                .allowUnassociatedTargets(true)
                .duration(2)
                .cutoff(1)
                .schedule("cron(0 10 ? * MON-FRI *)")
                .build();

        CompletableFuture<CreateMaintenanceWindowResponse> future =
getAsyncClient().createMaintenanceWindow(request);
        final String[] windowId = {null};
        future.whenComplete((response, ex) -> {
            if (response != null) {
                String maintenanceWindowId = response.windowId();
                System.out.println("The maintenance window id is " +
maintenanceWindowId);
                windowId[0] = maintenanceWindowId;
            } else {
                Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
                if (cause instanceof DocumentAlreadyExistsException) {
                    throw new CompletionException(cause);
                } else if (cause instanceof SsmException) {
                    throw new CompletionException(cause);
                } else {
                    throw new RuntimeException(cause);
                }
            }
        }).join();

        if (windowId[0] == null) {
            MaintenanceWindowFilter filter = MaintenanceWindowFilter.builder()
                    .key("name")
                    .values(winName)
                    .build();

            DescribeMaintenanceWindowsRequest winRequest =
DescribeMaintenanceWindowsRequest.builder()
                    .filters(filter)
                    .build();

            CompletableFuture<DescribeMaintenanceWindowsResponse> describeFuture =
getAsyncClient().describeMaintenanceWindows(winRequest);
            describeFuture.whenComplete((describeResponse, describeEx) -> {
                if (describeResponse != null) {
```

```
                    List<MaintenanceWindowIdentity> windows =
 describeResponse.windowIdentities();
                    if (!windows.isEmpty()) {
                        windowId[0] = windows.get(0).windowId();
                        System.out.println("Window ID: " + windowId[0]);
                    } else {
                        System.out.println("Window not found.");
                        windowId[0] = "";
                    }
                } else {
                    Throwable describeCause = (describeEx instanceof
 CompletionException) ? describeEx.getCause() : describeEx;
                    throw new RuntimeException("Error describing maintenance
 windows: " + describeCause.getMessage(), describeCause);
                }
            }).join();
        }

        return windowId[0];
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。
  - CreateDocument
  - CreateMaintenanceWindow
  - CreateOpsItem
  - DeleteMaintenanceWindow
  - ListCommandInvocations
  - SendCommand
  - UpdateOpsItem

操作

`CreateDocument`

以下代码示例演示了如何使用 `CreateDocument`。

适用于 Java 的 SDK 2.x

> **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。

```
/**
 * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateDocument中的。

`CreateMaintenanceWindow`

以下代码示例演示了如何使用 `CreateMaintenanceWindow`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateMaintenanceWindow中的。

`CreateOpsItem`

以下代码示例演示了如何使用 `CreateOpsItem`。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateOpsItem中的。

`DeleteDocument`

以下代码示例演示了如何使用 `DeleteDocument`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteDocument中的。

`DeleteMaintenanceWindow`

以下代码示例演示了如何使用 `DeleteMaintenanceWindow`。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteMaintenanceWindow中的。

DescribeOpsItems

以下代码示例演示了如何使用 DescribeOpsItems。

适用于 Java 的 SDK 2.x

> **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。

```
    /**
     * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeOpsItems中的。

`DescribeParameters`

以下代码示例演示了如何使用 `DescribeParameters`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeParameters中的。

`PutParameter`

以下代码示例演示了如何使用 `PutParameter`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考PutParameter中的。

SendCommand

以下代码示例演示了如何使用 SendCommand。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考SendCommand中的。

UpdateMaintenanceWindow

以下代码示例演示了如何使用 UpdateMaintenanceWindow。

适用于 Java 的 SDK 2.x

> **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。

```
   /**
    * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[UpdateMaintenanceWindow](#)中的。

`UpdateOpsItem`

以下代码示例演示了如何使用 `UpdateOpsItem`。

适用于 Java 的 SDK 2.x

> **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。

```
/**
 * 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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考UpdateOpsItem中的。

使用 SDK for 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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从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

`AnalyzeDocument`

以下代码示例演示了如何使用 `AnalyzeDocument`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考AnalyzeDocument中的。

`DetectDocumentText`

以下代码示例演示了如何使用 `DetectDocumentText`。

适用于 Java 的 SDK 2.x

> **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。

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(textractClient, sourceDoc);
        textractClient.close();
    }

    public static void detectDocText(TextractClient textractClient, String
 sourceDoc) {
        try {
            InputStream sourceStream = new FileInputStream(new File(sourceDoc));
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);

            // Get the input Document object as bytes.
            Document myDoc = Document.builder()
                    .bytes(sourceBytes)
                    .build();

            DetectDocumentTextRequest detectDocumentTextRequest =
 DetectDocumentTextRequest.builder()
                    .document(myDoc)
                    .build();

            // Invoke the Detect operation.
            DetectDocumentTextResponse textResponse =
 textractClient.detectDocumentText(detectDocumentTextRequest);
            List<Block> docInfo = textResponse.blocks();
            for (Block block : docInfo) {
                System.out.println("The block type is " +
 block.blockType().toString());
            }

            DocumentMetadata documentMetadata = textResponse.documentMetadata();
            System.out.println("The number of pages in the document is " +
 documentMetadata.pages());

        } catch (TextractException | FileNotFoundException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

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(textractClient, bucketName, docName);
        textractClient.close();
    }

    public static void detectDocTextS3(TextractClient textractClient, String
 bucketName, String docName) {
        try {
            S3Object s3Object = S3Object.builder()
                    .bucket(bucketName)
                    .name(docName)
                    .build();

            // Create a Document object and reference the s3Object instance.
            Document myDoc = Document.builder()
                    .s3Object(s3Object)
                    .build();

            DetectDocumentTextRequest detectDocumentTextRequest =
 DetectDocumentTextRequest.builder()
                    .document(myDoc)
                    .build();

            DetectDocumentTextResponse textResponse =
 textractClient.detectDocumentText(detectDocumentTextRequest);
            for (Block block : textResponse.blocks()) {
                System.out.println("The block type is " +
 block.blockType().toString());
            }

            DocumentMetadata documentMetadata = textResponse.documentMetadata();
            System.out.println("The number of pages in the document is " +
 documentMetadata.pages());

        } catch (TextractException e) {

            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DetectDocumentText中的。

`StartDocumentAnalysis`

以下代码示例演示了如何使用 `StartDocumentAnalysis`。

适用于 Java 的 SDK 2.x

> **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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.textract.model.S3Object;
import software.amazon.awssdk.services.textract.TextractClient;
import software.amazon.awssdk.services.textract.model.StartDocumentAnalysisRequest;
import software.amazon.awssdk.services.textract.model.DocumentLocation;
import software.amazon.awssdk.services.textract.model.TextractException;
import software.amazon.awssdk.services.textract.model.StartDocumentAnalysisResponse;
import software.amazon.awssdk.services.textract.model.GetDocumentAnalysisRequest;
import software.amazon.awssdk.services.textract.model.GetDocumentAnalysisResponse;
import software.amazon.awssdk.services.textract.model.FeatureType;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class StartDocumentAnalysis {
    public static void main(String[] args) {
        final String usage = """

                Usage:
```

```
                    <bucketName> <docName>\s

                Where:
                    bucketName - The name of the Amazon S3 bucket that contains the
document.\s
                    docName - The document name (must be an image, for example,
book.png).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String docName = args[1];
        Region region = Region.US_WEST_2;
        TextractClient textractClient = TextractClient.builder()
                .region(region)
                .build();

        String jobId = startDocAnalysisS3(textractClient, bucketName, docName);
        System.out.println("Getting results for job " + jobId);
        String status = getJobResults(textractClient, jobId);
        System.out.println("The job status is " + status);
        textractClient.close();
    }

    public static String startDocAnalysisS3(TextractClient textractClient, String
bucketName, String docName) {
        try {
            List<FeatureType> myList = new ArrayList<>();
            myList.add(FeatureType.TABLES);
            myList.add(FeatureType.FORMS);

            S3Object s3Object = S3Object.builder()
                    .bucket(bucketName)
                    .name(docName)
                    .build();

            DocumentLocation location = DocumentLocation.builder()
                    .s3Object(s3Object)
                    .build();
```

```
            StartDocumentAnalysisRequest documentAnalysisRequest =
StartDocumentAnalysisRequest.builder()
                    .documentLocation(location)
                    .featureTypes(myList)
                    .build();

            StartDocumentAnalysisResponse response =
textractClient.startDocumentAnalysis(documentAnalysisRequest);

            // Get the job ID
            String jobId = response.jobId();
            return jobId;

        } catch (TextractException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    private static String getJobResults(TextractClient textractClient, String jobId)
{
        boolean finished = false;
        int index = 0;
        String status = "";

        try {
            while (!finished) {
                GetDocumentAnalysisRequest analysisRequest =
GetDocumentAnalysisRequest.builder()
                        .jobId(jobId)
                        .maxResults(1000)
                        .build();

                GetDocumentAnalysisResponse response =
textractClient.getDocumentAnalysis(analysisRequest);
                status = response.jobStatus().toString();

                if (status.compareTo("SUCCEEDED") == 0)
                    finished = true;
                else {
                    System.out.println(index + " status is: " + status);
                    Thread.sleep(1000);
                }
```

```
                index++;
            }

            return status;

        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartDocumentAnalysis中的。

场景

创建用于分析客户反馈的应用程序

以下代码示例说明如何创建应用程序来分析客户意见卡、翻译其母语、确定其情绪并根据译后的文本生成音频文件。

适用于 Java 的 SDK 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 SDK for 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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从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`

以下代码示例演示了如何使用 `ListTranscriptionJobs`。

适用于 Java 的 SDK 2.x

> **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。

```
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 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[ListTranscriptionJobs](#)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。
- 获取存储转录的 URI。

有关更多信息，请参阅 Amazon Transcribe 入门。

适用于 Java 的 SDK 2.x

> **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。

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);
                })
                .onComplete(() -> {
                    System.out.println("=== All records stream successfully ===");
                })
                .subscriber(event -> {
                    List<Result> results = ((TranscriptEvent)
 event).transcript().results();
                    if (results.size() > 0) {
                        if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                        }
                    }
                })
                .build();
    }


    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private static Subscription currentSubscription;
        private final InputStream inputStream;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
```

```
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (currentSubscription == null) {
                currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                currentSubscription.cancel();
                currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private final ExecutorService executor = Executors.newFixedThreadPool(1);
        private final AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
{
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
```

```
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》中的以下主题。

- GetTranscriptionJob
- StartTranscriptionJob

使用适用于 Java 的 SDK 2.x 的 Amazon Transcribe Streaming 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 Streaming 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的上下文查看操作。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从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

`StartMedicalStreamTranscription`

以下代码示例演示了如何使用 `StartMedicalStreamTranscription`。

适用于 Java 的 SDK 2.x

> **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。

```
/*
To run this AWS code example, ensure that you have set up your development
 environment, including your AWS credentials.
```

```
 For information, see this documentation topic:

 https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html

This code demonstrates the process of starting a medical transcription job using the
 AWS Transcribe
Streaming service, including setting up the audio input stream, configuring the
 transcription request,
and handling the transcription response.
 */

public class TranscribeMedicalStreamingDemoApp {
    private static TranscribeStreamingAsyncClient client;

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

        client = TranscribeStreamingAsyncClient.builder()
            .credentialsProvider(getCredentials())
            .build();

        CompletableFuture<Void> result =
 client.startMedicalStreamTranscription(getMedicalRequest(16_000),
            new AudioStreamPublisher(getStreamFromMic()),
            getMedicalResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromMic() throws LineUnavailableException {

        // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
        int sampleRate = 16000;
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Line not supported");
            throw new LineUnavailableException("The audio system microphone line is
 not supported.");
        }
```

```
        TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
        line.open(format);
        line.start();

        InputStream audioStream = new AudioInputStream(line);
        return audioStream;
    }

    private static AwsCredentialsProvider getCredentials() {
        return DefaultCredentialsProvider.create();
    }

    private static StartMedicalStreamTranscriptionRequest getMedicalRequest(Integer
mediaSampleRateHertz) {
        return StartMedicalStreamTranscriptionRequest.builder()
            .languageCode(LanguageCode.EN_US.toString()) // For medical
transcription, EN_US is typically used.
            .mediaEncoding(MediaEncoding.PCM)
            .mediaSampleRateHertz(mediaSampleRateHertz)
            .specialty(Specialty.PRIMARYCARE) // Specify the medical specialty.
            .type(Type.CONVERSATION) // Set the type as CONVERSATION or DICTATION.
            .build();
    }

    private static StartMedicalStreamTranscriptionResponseHandler
getMedicalResponseHandler() {
        return StartMedicalStreamTranscriptionResponseHandler.builder()
            .onResponse(r -> {
                System.out.println("Received Initial response");
            })
            .onError(e -> {
                System.out.println(e.getMessage());
                StringWriter sw = new StringWriter();
                e.printStackTrace(new PrintWriter(sw));
                System.out.println("Error Occurred: " + sw.toString());
            })
            .onComplete(() -> {
                System.out.println("=== All records streamed successfully ===");
            })
            .subscriber(event -> {
                List<MedicalResult> results = ((MedicalTranscriptEvent)
event).transcript().results();
                if (results.size() > 0) {
```

```
                    if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                    }
                }
            })
            .build();
    }

    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private final InputStream inputStream;
        private static Subscription currentSubscription;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (this.currentSubscription == null) {
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                this.currentSubscription.cancel();
                this.currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024 * 1;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private ExecutorService executor = Executors.newFixedThreadPool(1);
        private AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
 {
            this.subscriber = s;
            this.inputStream = inputStream;
        }
```

```
        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);
            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
```

```
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                .audioChunk(SdkBytes.fromByteBuffer(bb))
                .build();
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartMedicalStreamTranscription中的。

`StartStreamTranscription`

以下代码示例演示了如何使用 `StartStreamTranscription`。

适用于 Java 的 SDK 2.x

> **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。

```
public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
    private static TranscribeStreamingAsyncClient client;

    public static void main(String[] args)
            throws URISyntaxException, ExecutionException, InterruptedException,
 LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
```

```
                .credentialsProvider(getCredentials())
                .region(REGION)
                .build();

        CompletableFuture<Void> result =
client.startStreamTranscription(getRequest(16_000),
                new AudioStreamPublisher(getStreamFromMic()),
                getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromMic() throws LineUnavailableException {

        // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
        int sampleRate = 16000;
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Line not supported");
            System.exit(0);
        }

        TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
        line.open(format);
        line.start();

        InputStream audioStream = new AudioInputStream(line);
        return audioStream;
    }

    private static AwsCredentialsProvider getCredentials() {
        return DefaultCredentialsProvider.create();
    }

    private static StartStreamTranscriptionRequest getRequest(Integer
mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
                .languageCode(LanguageCode.EN_US.toString())
                .mediaEncoding(MediaEncoding.PCM)
                .mediaSampleRateHertz(mediaSampleRateHertz)
                .build();
```

```
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
                .onResponse(r -> {
                    System.out.println("Received Initial response");
                })
                .onError(e -> {
                    System.out.println(e.getMessage());
                    StringWriter sw = new StringWriter();
                    e.printStackTrace(new PrintWriter(sw));
                    System.out.println("Error Occurred: " + sw);
                })
                .onComplete(() -> {
                    System.out.println("=== All records stream successfully ===");
                })
                .subscriber(event -> {
                    List<Result> results = ((TranscriptEvent)
 event).transcript().results();
                    if (results.size() > 0) {
                        if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                        }
                    }
                })
                .build();
    }


    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private static Subscription currentSubscription;
        private final InputStream inputStream;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (currentSubscription == null) {
                currentSubscription = new SubscriptionImpl(s, inputStream);
```

```
            } else {
                currentSubscription.cancel();
                currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private final ExecutorService executor = Executors.newFixedThreadPool(1);
        private final AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
{
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
```

```
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartStreamTranscription中的。

场景

转录音频文件

以下代码示例演示如何使用 Amazon Transcribe 流式转录来生成源音频文件的转录。

适用于 Java 的 SDK 2.x

> **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。

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

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartStreamTranscription中的。

转录来自麦克风的音频

以下代码示例演示如何使用 Amazon Transcribe 流式转录从麦克风生成转录。

适用于 Java 的 SDK 2.x

> **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。

```
public class TranscribeStreamingDemoApp {
    private static final Region REGION = Region.US_EAST_1;
```

```
    private static TranscribeStreamingAsyncClient client;

    public static void main(String[] args)
            throws URISyntaxException, ExecutionException, InterruptedException,
LineUnavailableException {

        client = TranscribeStreamingAsyncClient.builder()
                .credentialsProvider(getCredentials())
                .region(REGION)
                .build();

        CompletableFuture<Void> result =
client.startStreamTranscription(getRequest(16_000),
                new AudioStreamPublisher(getStreamFromMic()),
                getResponseHandler());

        result.get();
        client.close();
    }

    private static InputStream getStreamFromMic() throws LineUnavailableException {

        // Signed PCM AudioFormat with 16kHz, 16 bit sample size, mono
        int sampleRate = 16000;
        AudioFormat format = new AudioFormat(sampleRate, 16, 1, true, false);
        DataLine.Info info = new DataLine.Info(TargetDataLine.class, format);

        if (!AudioSystem.isLineSupported(info)) {
            System.out.println("Line not supported");
            System.exit(0);
        }

        TargetDataLine line = (TargetDataLine) AudioSystem.getLine(info);
        line.open(format);
        line.start();

        InputStream audioStream = new AudioInputStream(line);
        return audioStream;
    }

    private static AwsCredentialsProvider getCredentials() {
        return DefaultCredentialsProvider.create();
    }
```

```
    private static StartStreamTranscriptionRequest getRequest(Integer
 mediaSampleRateHertz) {
        return StartStreamTranscriptionRequest.builder()
                .languageCode(LanguageCode.EN_US.toString())
                .mediaEncoding(MediaEncoding.PCM)
                .mediaSampleRateHertz(mediaSampleRateHertz)
                .build();
    }

    private static StartStreamTranscriptionResponseHandler getResponseHandler() {
        return StartStreamTranscriptionResponseHandler.builder()
                .onResponse(r -> {
                    System.out.println("Received Initial response");
                })
                .onError(e -> {
                    System.out.println(e.getMessage());
                    StringWriter sw = new StringWriter();
                    e.printStackTrace(new PrintWriter(sw));
                    System.out.println("Error Occurred: " + sw);
                })
                .onComplete(() -> {
                    System.out.println("=== All records stream successfully ===");
                })
                .subscriber(event -> {
                    List<Result> results = ((TranscriptEvent)
 event).transcript().results();
                    if (results.size() > 0) {
                        if (!
results.get(0).alternatives().get(0).transcript().isEmpty()) {

 System.out.println(results.get(0).alternatives().get(0).transcript());
                        }
                    }
                })
                .build();
    }


    private static class AudioStreamPublisher implements Publisher<AudioStream> {
        private static Subscription currentSubscription;
        private final InputStream inputStream;

        private AudioStreamPublisher(InputStream inputStream) {
            this.inputStream = inputStream;
```

```
        }

        @Override
        public void subscribe(Subscriber<? super AudioStream> s) {

            if (currentSubscription == null) {
                currentSubscription = new SubscriptionImpl(s, inputStream);
            } else {
                currentSubscription.cancel();
                currentSubscription = new SubscriptionImpl(s, inputStream);
            }
            s.onSubscribe(currentSubscription);
        }
    }

    public static class SubscriptionImpl implements Subscription {
        private static final int CHUNK_SIZE_IN_BYTES = 1024;
        private final Subscriber<? super AudioStream> subscriber;
        private final InputStream inputStream;
        private final ExecutorService executor = Executors.newFixedThreadPool(1);
        private final AtomicLong demand = new AtomicLong(0);

        SubscriptionImpl(Subscriber<? super AudioStream> s, InputStream inputStream)
{
            this.subscriber = s;
            this.inputStream = inputStream;
        }

        @Override
        public void request(long n) {
            if (n <= 0) {
                subscriber.onError(new IllegalArgumentException("Demand must be
positive"));
            }

            demand.getAndAdd(n);

            executor.submit(() -> {
                try {
                    do {
                        ByteBuffer audioBuffer = getNextEvent();
                        if (audioBuffer.remaining() > 0) {
                            AudioEvent audioEvent =
audioEventFromBuffer(audioBuffer);
```

```
                            subscriber.onNext(audioEvent);
                        } else {
                            subscriber.onComplete();
                            break;
                        }
                    } while (demand.decrementAndGet() > 0);
                } catch (Exception e) {
                    subscriber.onError(e);
                }
            });
        }

        @Override
        public void cancel() {
            executor.shutdown();
        }

        private ByteBuffer getNextEvent() {
            ByteBuffer audioBuffer = null;
            byte[] audioBytes = new byte[CHUNK_SIZE_IN_BYTES];

            int len = 0;
            try {
                len = inputStream.read(audioBytes);

                if (len <= 0) {
                    audioBuffer = ByteBuffer.allocate(0);
                } else {
                    audioBuffer = ByteBuffer.wrap(audioBytes, 0, len);
                }
            } catch (IOException e) {
                throw new UncheckedIOException(e);
            }

            return audioBuffer;
        }

        private AudioEvent audioEventFromBuffer(ByteBuffer bb) {
            return AudioEvent.builder()
                    .audioChunk(SdkBytes.fromByteBuffer(bb))
                    .build();
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartStreamTranscription中的。

使用适用于 Java 的 SDK 2.x 的 Amazon Translate 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 Translate 配合使用来执行操作和实现常见场景。

场景是向您演示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

每个示例都包含一个指向完整源代码的链接，您可以从中找到有关如何在上下文中设置和运行代码的说明。

主题

- 场景

场景

构建 Amazon Lex 聊天机器人

以下代码示例演示如何创建用于吸引网站访客的聊天机器人。

适用于 Java 的 SDK 2.x

展示如何使用 Amazon Lex API 在 Web 应用程序中创建聊天机器人，以吸引网站访客。

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 Amazon SNS 应用程序

以下代码示例演示如何创建具有订阅和发布功能并能转换消息的应用程序。

适用于 Java 的 SDK 2.x

展示如何使用 Amazon Simple Notification Service 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 API 的示例的完整源代码和说明，请参阅上的GitHub完整示例。

本示例中使用的服务

- Amazon SNS
- Amazon Translate

创建用于分析客户反馈的应用程序

以下代码示例说明如何创建应用程序来分析客户意见卡、翻译其母语、确定其情绪并根据译后的文本生成音频文件。

适用于 Java 的 SDK 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

从AWS SDK for Java 的 1.x 版迁移到 2.x 版

AWS SDK for Java 2.x 是对基于 Java 8+ 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
- 如何将代码从AWS SDK for Java 1.x 迁移到 2.x
- AWS SDK for Java 1.x 和 2.x 之间的差异
- 并行使用适用于 Java 的 SDK 1.x 和 2.x 版
- 查找使用AWS SDK for Java 1.x 客户端的应用程序

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 函数的开发工具包启动时间性能已提高。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。

- 快速入门
- 设置
- AWS SDK for Java 2.x 代码示例
- 使用 SDK

- AWS SDK for Java 的安全性

如何将代码从AWS SDK for Java 1.x 迁移到 2.x

您可以通过几种方式迁移现有适用于 Java 的 SDK 1.x 应用程序。

1. 使用迁移工具实现自动化方法。
2. 手动方法，将 1.x 导入逐步替换为 2.x 导入。

我们建议您先从使用迁移工具开始。该工具能自动执行从 1.x 到 2.x 代码迁移中的大量重复性的替换工作。

由于该工具不会迁移所有功能，因此您需要在运行该工具后搜索剩余的 v1 代码。当您发现该工具未迁移的代码时，请按照分步说明（手动方法）进行操作，并按照迁移指南文章完成迁移。

主题

- AWS SDK for Java 迁移工具
- 带示例的迁移分步说明

AWS SDK for Java 迁移工具

AWS SDK for Java 提供了一个迁移工具，可协助自动将适用于 Java 的 SDK 1.x（V1）代码迁移到 2.x（V2）。该工具使用 OpenRewrite（一个用于重构源代码的开源工具）来执行迁移。OpenRewrite 使用代码修改规则（称为“配方”）自动将源代码从 V1 更新为 V2 语法和模式。

该工具支持 SDK 服务客户端和 S3 Transfer Manager 高级库的代码修改规则。不支持其他高级别 API（例如 V1 的 `DynamoDBMapper` 到 V2 的 DynamoDB 增强型客户端 API）的代码修改规则。

有关限制的更多详细信息，请参阅本页末尾。有关使用手动迁移步骤的常见不受支持代码模式的详细示例，请参阅不支持的代码模式。

使用迁移工具

迁移 Maven 项目

按照以下说明，使用 OpenRewrite Maven 插件工具迁移基于 Maven 的适用于 Java 的 SDK 1.x 项目。

1. 如果需要，导航到 Maven 项目的根目录

   打开终端（命令行）窗口并导航到基于 Maven 的应用程序的根目录。

2. 运行插件的 `rewrite-maven-plugin` 命令

   您可以从两种模式（Maven 目标）中进行选择：`dryRun` 和 `run`。

   `dryRun` 模式

   在 `dryRun` 模式下，插件会在控制台输出中生成差异日志，并在 `target/rewrite` 文件夹中生成名为 `rewrite.patch` 的补丁文件。借助此模式，您可以预览将要进行的更改，因为这不会对源代码文件进行任何更改。

   以下示例说明如何在 `dryRun` 模式下调用插件。

```
mvn org.openrewrite.maven:rewrite-maven-plugin:<rewrite-plugin-version>*:dryRun \
  -Drewrite.recipeArtifactCoordinates=software.amazon.awssdk:v2-
migration:<sdkversion>** \
  -Drewrite.activeRecipes=software.amazon.awssdk.v2migration.AwsSdkJavaV1ToV2
```

   *将 *<rewrite-plugin-version>* 替换为您在此测试文件中看到的 `rewriteMavenPluginVersion` 值。

   **将 *<sdkversion>* 替换为 2.x SDK 版本。访问 Maven Central 以检查最新版本。

   > **Important**
   >
   > 因为其他版本可能无法运行，请务必使用测试文件中显示的 `rewrite-maven-plugin` 版本。

   `dryRun` 模式下的控制台输出应类似于以下输出。

```
[WARNING] These recipes would make changes to project/src/test/resources/maven/
before/pom.xml:
[WARNING]     software.amazon.awssdk.v2migration.AwsSdkJavaV1ToV2
[WARNING]         software.amazon.awssdk.v2migration.UpgradeSdkDependencies
[WARNING]             org.openrewrite.java.dependencies.AddDependency:
 {groupId=software.amazon.awssdk, artifactId=apache-client, version=2.27.0,
 onlyIfUsing=com.amazonaws.ClientConfiguration}
```

```
[WARNING]             org.openrewrite.java.dependencies.AddDependency:
 {groupId=software.amazon.awssdk, artifactId=netty-nio-client, version=2.27.0,
 onlyIfUsing=com.amazonaws.ClientConfiguration}
[WARNING]             org.openrewrite.java.dependencies.ChangeDependency:
 {oldGroupId=com.amazonaws, oldArtifactId=aws-java-sdk-bom,
 newGroupId=software.amazon.awssdk, newArtifactId=bom, newVersion=2.27.0}
[WARNING]             org.openrewrite.java.dependencies.ChangeDependency:
 {oldGroupId=com.amazonaws, oldArtifactId=aws-java-sdk-s3,
 newGroupId=software.amazon.awssdk, newArtifactId=s3, newVersion=2.27.0}
[WARNING]             org.openrewrite.java.dependencies.ChangeDependency:
 {oldGroupId=com.amazonaws, oldArtifactId=aws-java-sdk-sqs,
 newGroupId=software.amazon.awssdk, newArtifactId=sqs, newVersion=2.27.0}
[WARNING] These recipes would make changes to project/src/test/resources/maven/
before/src/main/java/foo/bar/Application.java:
[WARNING]     software.amazon.awssdk.v2migration.AwsSdkJavaV1ToV2
[WARNING]         software.amazon.awssdk.v2migration.S3GetObjectConstructorToFluent
[WARNING]             software.amazon.awssdk.v2migration.ConstructorToFluent
[WARNING]         software.amazon.awssdk.v2migration.S3StreamingResponseToV2
[WARNING]         software.amazon.awssdk.v2migration.ChangeSdkType
[WARNING]         software.amazon.awssdk.v2migration.ChangeSdkCoreTypes
[WARNING]             software.amazon.awssdk.v2migration.ChangeExceptionTypes
[WARNING]                 org.openrewrite.java.ChangeType:
 {oldFullyQualifiedTypeName=com.amazonaws.AmazonClientException,
 newFullyQualifiedTypeName=software.amazon.awssdk.core.exception.SdkException}
[WARNING]                 org.openrewrite.java.ChangeMethodName:
 {methodPattern=com.amazonaws.AmazonServiceException getRequestId(),
 newMethodName=requestId}
[WARNING]                 org.openrewrite.java.ChangeMethodName:
 {methodPattern=com.amazonaws.AmazonServiceException getErrorCode(),
 newMethodName=awsErrorDetails().errorCode}
[WARNING]                 org.openrewrite.java.ChangeMethodName:
 {methodPattern=com.amazonaws.AmazonServiceException getServiceName(),
 newMethodName=awsErrorDetails().serviceName}
[WARNING]                 org.openrewrite.java.ChangeMethodName:
 {methodPattern=com.amazonaws.AmazonServiceException getErrorMessage(),
 newMethodName=awsErrorDetails().errorMessage}
[WARNING]                 org.openrewrite.java.ChangeMethodName:
 {methodPattern=com.amazonaws.AmazonServiceException getRawResponse(),
 newMethodName=awsErrorDetails().rawResponse().asByteArray}
[WARNING]                 org.openrewrite.java.ChangeMethodName:
 {methodPattern=com.amazonaws.AmazonServiceException getRawResponseContent(),
 newMethodName=awsErrorDetails().rawResponse().asUtf8String}
```

```
[WARNING]                 org.openrewrite.java.ChangeType:
 {oldFullyQualifiedTypeName=com.amazonaws.AmazonServiceException,
 newFullyQualifiedTypeName=software.amazon.awssdk.awscore.exception.AwsServiceException}
[WARNING]         software.amazon.awssdk.v2migration.NewClassToBuilderPattern
[WARNING]             software.amazon.awssdk.v2migration.NewClassToBuilder
[WARNING]             software.amazon.awssdk.v2migration.V1SetterToV2
[WARNING]         software.amazon.awssdk.v2migration.V1GetterToV2
...
[WARNING]         software.amazon.awssdk.v2migration.V1BuilderVariationsToV2Builder
[WARNING]         software.amazon.awssdk.v2migration.NewClassToBuilderPattern
[WARNING]             software.amazon.awssdk.v2migration.NewClassToBuilder
[WARNING]             software.amazon.awssdk.v2migration.V1SetterToV2
[WARNING]         software.amazon.awssdk.v2migration.HttpSettingsToHttpClient
[WARNING]         software.amazon.awssdk.v2migration.WrapSdkClientBuilderRegionStr
[WARNING] Patch file available:
[WARNING]     project/src/test/resources/maven/before/target/rewrite/rewrite.patch
[WARNING] Estimate time saved: 20m
[WARNING] Run 'mvn rewrite:run' to apply the recipes.
```

run 模式

当您在 run 模式下运行插件时，它会修改磁盘上的源代码以应用更改。运行命令之前，确保您有源代码的备份。

以下示例说明如何在 run 模式下调用插件。

```
mvn org.openrewrite.maven:rewrite-maven-plugin:<rewrite-plugin-version>*:run \
  -Drewrite.recipeArtifactCoordinates=software.amazon.awssdk:v2-
migration:<sdkversion>** \
  -Drewrite.activeRecipes=software.amazon.awssdk.v2migration.AwsSdkJavaV1ToV2
```

*将 *<rewrite-plugin-version>* 替换为您在此测试文件中看到的 rewriteMavenPluginVersionvalue。

**将 *<sdkversion>* 替换为 2.x SDK 版本。访问 Maven Central 以检查最新版本。

运行命令后，编译应用程序并运行测试以验证更改。

迁移 Gradle 项目

按照以下说明，使用 OpenRewrite Gradle 插件工具迁移基于 Gradle 的适用于 Java 的 SDK 1.x 项目。

1. 如果需要，导航到 Gradle 项目的根目录

   打开终端（命令行）窗口并导航到基于 Gradle 的应用程序的根目录。

2. 创建 Gradle 初始化脚本

   在目录中创建一个包含以下内容的 `init.gradle` 文件。

```
initscript {
    repositories {
        maven { url "https://plugins.gradle.org/m2" }
    }
    dependencies {
        classpath("org.openrewrite:plugin:<rewrite-plugin-version>*")
    }
}

rootProject {
    plugins.apply(org.openrewrite.gradle.RewritePlugin)
    dependencies {
        rewrite("software.amazon.awssdk:v2-migration:latest.release")
    }

    afterEvaluate {
        if (repositories.isEmpty()) {
            repositories {
                mavenCentral()
            }
        }
    }
}
```

   *将 *<rewrite-plugin-version>* 替换为您在此测试文件中看到的版本。

3. 运行 `rewrite` 命令

   与 Maven 插件一样，您可以在 `dryRun` 或 `run` 模式下运行 Gradle 插件。

   `dryRun` 模式

以下示例说明如何在 dryRun 模式下调用插件。

```
gradle rewriteDryRun --init-script init.gradle \
  -Drewrite.activeRecipes=software.amazon.awssdk.v2migration.AwsSdkJavaV1ToV2
```

run 模式

以下示例说明如何在 run 模式下调用插件。

```
gradle rewriteRun --init-script init.gradle \
  -Drewrite.activeRecipes=software.amazon.awssdk.v2migration.AwsSdkJavaV1ToV2
```

目前的局限性

虽然通过更新到 V2 等效项的代码修改规则可让迁移支持大多数 V1 代码，但有些类和方法并未包括在内。对于这些类和方法，请按照分步说明手动迁移代码。

对于某些不支持的代码修改规则，迁移工具可能会添加以下内容开头的注释：

```
/*AWS SDK for Java v2 migration: Transform for ...
```

在注释之后，该工具会输出方法或类的 V2 版本的通用存根。例如，在以下输出中，迁移工具尝试迁移 V1 S3 客户端的 setBucketLifecycleConfiguration 方法：

```
/*AWS SDK for Java v2 migration: Transform for setBucketLifecycleConfiguration method
 not supported.
Please manually migrate your code by using builder pattern, update from
 BucketLifecycleConfiguration.Rule
to LifecycleRule, StorageClass to TransitionStorageClass, and adjust imports and
 names.*/
s3.putBucketLifecycleConfiguration(
        PutBucketLifecycleConfigurationRequest.builder()
            .bucket(bucketName)
            .lifecycleConfiguration(BucketLifecycleConfiguration.builder()
                .build())
            .build());
```

以下列表中的链接可带您前往查看迁移信息，以帮助您手动迁移代码。

- the section called “S3 客户端差异”
- S3 Transfer Manager（TransferManager）
- DynamoDB 对象映射（DynamoDBMapper）
- EC2 元数据实用程序（EC2MetadataUtils）
- Waiter（AmazonDynamoDBWaiters）
- IAM 策略生成器（策略）
- CloudFront 预签名（CloudFrontUrlSigner、CloudFrontCookieSigner）
- S3 事件通知（S3EventNotification）
- SDK 指标发布（1.x 文档、2.x 文档）
- 不支持的代码模式 - 需要手动迁移的常见代码模式的详细示例

迁移工具不支持的代码模式

迁移工具会自动将大多数 v1 代码转换为 v2。但是，某些代码模式需要手动迁移。本主题提供了最常见不受支持模式的详细示例，并向您展示了如何手动转换这些模式。

以下模式列表并非详尽内容。如果运行迁移工具后您的代码无法编译，请按照分步迁移说明手动迁移剩余的 v1 代码。

带参数的请求对象构造函数

对于请求 POJO（不包括 Amazon S3），迁移工具仅转换 setter 方法。该工具不支持带参数的构造函数。

支持的模式：使用 setter 请求对象（没有构造函数参数）

之前（原始 v1 代码）：

```
import com.amazonaws.services.sqs.model.SendMessageRequest;

SendMessageRequest request = new SendMessageRequest().withMessageBody("Hello World");
request.setQueueUrl("https://sqs.us-west-2.amazonaws.com/0123456789012/demo-queue");
```

之后（迁移工具结果）：

```
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;

SendMessageRequest request = SendMessageRequest.builder()
```

```
    .messageBody("Hello World").build();
request = request.toBuilder()
    .queueUrl("https://sqs.us-west-2.amazonaws.com/0123456789012/demo-queue").build();
```

不支持的模式：带参数的请求对象构造函数

迁移工具无法转换带参数的构造函数：

在手动迁移之前，但在使用迁移工具之后：

```
import software.amazon.awssdk.services.sqs.model.SendMessageRequest; // Import updated
 to v2.

// This pattern requires manual migration.
SendMessageRequest request = new SendMessageRequest(
    "https://sqs.us-west-2.amazonaws.com/0123456789012/demo-queue",
    "Hello World");
```

迁移工具将导入转换为 v2，但构造函数代码保持不变，需要手动更新才能使用生成器模式。

手动迁移后：

```
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;

SendMessageRequest request = SendMessageRequest.builder()
    .messageBody("Hello World")
    .queueUrl("https://sqs.us-west-2.amazonaws.com/0123456789012/demo-queue")
    .build();
```

带有单独参数的服务客户端方法

迁移工具无法转换采用单个参数而非请求对象（不包括 Amazon S3）的服务客户端方法。

之前（v1 代码）：

```
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClient;

AmazonSQS sqs = new AmazonSQSClient();
// The following v1 method takes individual parameters.
sqs.sendMessage("https://sqs.us-west-2.amazonaws.com/0123456789012/demo-queue", "Hello
 World");
```

之后（迁移工具结果，不编译）：

```
import software.amazon.awssdk.services.sqs.SqsClient;  // Import updated to v2.
// No import statement for the v2 request POJO.

SqsClient sqs = SqsClient.builder().build();

// Does not compile–v2 methods only accept request POJOs.
sqs.sendMessage("https://sqs.us-west-2.amazonaws.com/0123456789012/demo-queue", "Hello
 World");
```

您必须手动更新方法参数才能使用请求对象：

```
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest; // Add manually.

SqsClient sqs = SqsClient.builder().build();

// Corrected v2 code.
SendMessageRequest request = SendMessageRequest.builder()
    .queueUrl("https://sqs.us-west-2.amazonaws.com/0123456789012/demo-queue")
    .messageBody("Hello World")
    .build();
sqs.sendMessage(request);
```

请求超时方法

迁移工具不会转换对请求对象设置超时的方法。

之前（v1 代码）：

```
import com.amazonaws.services.sqs.model.SendMessageRequest;

SendMessageRequest request = new SendMessageRequest();
request.setSdkRequestTimeout(7);
```

之后（迁移工具结果，不编译）：

```
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;  // Import updated
 to v2.

SendMessageRequest request = SendMessageRequest.builder().build();
```

```
// Does not compile.
request.setSdkRequestTimeout(7);
```

您必须手动迁移才能使用 v2 的 overrideConfiguration 方法：

```
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;
import java.time.Duration;

SendMessageRequest request = SendMessageRequest.builder().build();

// Corrected v2 code.
request = request.toBuilder()
    .overrideConfiguration(o -> o.apiCallTimeout(Duration.ofSeconds(7)))
    .build();
```

带参数的服务客户端构造函数

迁移工具转换空的服务客户端构造函数，但无法转换接受凭证或配置等参数的构造函数。

之前（v1 代码）：

```
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClient;

AWSCredentials awsCredentials = new BasicAWSCredentials("akid", "skid");
AmazonSQS sqs = new AmazonSQSClient(awsCredentials);
```

之后（迁移工具结果，不编译）：

```
import software.amazon.awssdk.auth.credentials.AwsCredentials;
import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;
import software.amazon.awssdk.services.sqs.SqsClient;  // Import updated to v2.

AwsCredentials awsCredentials = AwsBasicCredentials.create("akid", "skid");

// Does not compile.
SqsClient sqs = new SqsClient(awsCredentials);
```

必须手动更新服务客户端构造函数才能使用生成器模式：

```
import software.amazon.awssdk.auth.credentials.AwsCredentials;
import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;
import software.amazon.awssdk.auth.credentials.StaticCredentialsProvider;  // Add
 manually.
import software.amazon.awssdk.services.sqs.SqsClient;

AwsCredentials awsCredentials = AwsBasicCredentials.create("akid", "skid");

// Corrected v2 code.
SqsClient sqs = SqsClient.builder()
    .credentialsProvider(StaticCredentialsProvider.create(awsCredentials))
    .build();
```

带示例的迁移分步说明

本部分提供了分步指南，说明如何将当前使用适用于 Java 的 SDK v1.x 的应用程序迁移到适用于 Java 的 SDK 2.x。第一部分概述了各个步骤，然后是详细的迁移示例。

此处介绍的步骤说明了正常使用案例的迁移，应用程序使用模型驱动的服务客户端调用 AWS 服务。如果您需要迁移使用更高级别 API（例如 S3 Transfer Manager 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。

此处描述的方法仅为建议。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 的 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 v2 依赖项都来自同一个版本。您的 POM 可以同时包含 v1 和 v2 依赖项。这样您就可以通过增量方式迁移代码，而不必一次全部更改。

适用于 Java 的 SDK 2.x BOM

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

```
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

您可以在 Maven Central 存储库中找到最新版本。

2. 在文件中搜索 v1 类导入语句

通过在应用程序的文件中扫描 v1 导入中使用的 SERVICE_ID，您可以找到所使用的唯一 SERVICE_ID。SERVICE_ID 是 AWS 服务的简短、唯一名称。例如，`cognitoidentity` 是 Amazon Cognito Identity 的 SERVICE_ID。

3. 从 v1 导入语句中确定 v2 Maven 依赖项

找到所有唯一的 v1 SERVICE_ID 后，您可以通过参考 the section called “软件包名称到 artifactId 的映射” 来确定 v2 依赖项的相应 Maven 构件。

4. 将 v2 依赖项元素添加到 POM 文件中

使用步骤 3 中确定的依赖项元素更新 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，例如使用生成器而不是构造函数，以及使用 fluent getter 和 setter。

6. 从 POM 中移除 v1 Maven 依赖项，从文件中移除 v1 导入

将代码迁移到使用 v2 类后，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。

示例迁移

在此示例中，我们迁移了一个使用适用于 Java 的 SDK v1 并可以访问多个 AWS 服务的应用程序。我们在步骤 5 中详细解析了以下 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。

要迁移的 v1 方法

下面仅列出了来自 Java 文件的 v1 SDK 导入。

```
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 的 SDK 2.x 的 Maven BOM 添加到 POM 的 dependencyManagement 部分，与任何其他依赖项放在一起。如果您的 POM 文件中有 SDK v1 的 BOM，请暂时将其保留。您将在后面的步骤中将其删除。

在一开始就配置好 POM 依赖项管理

```
<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>2.27.21</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
```

```
</dependencyManagement>
```

2. 在文件中搜索 v1 类导入语句

在应用程序的代码中搜索唯一出现的 `import com.amazonaws.services`。这有助于我们确定项目使用的 v1 依赖项。如果您的应用程序有一个 Maven POM 文件列出了 v1 依赖项，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。

在代码库的根目录中执行以下 `ripgrep` 命令。`ripgrep` 找到导入语句后，会将它们通过管道传送到 `cut`、`sort` 和 `uniq` 命令，用于隔离 SERVICE_ID。

```
rg --no-filename 'import\s+com\.amazonaws\.services' | cut -d '.' -f 4 | sort | uniq
```

对于此应用程序，在控制台会记录以下 SERVICE_ID。

```
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。就我们当前的目的而言，具体的类名无关紧要。我们只需要找到使用的 SERVICE_ID。

```
com.amazonaws.services.autoscaling.*
com.amazonaws.services.cloudformation.*
com.amazonaws.services.ec2.*
com.amazonaws.services.identitymanagement.*
```

3. 从 v1 导入语句中确定 v2 Maven 依赖项

我们从步骤 2 中分离出来的 v1 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，因此您已拥有所需的信息，可以向 POM 文件添加依赖项块。

下表说明了我们如何确定 v2 依赖项。

v1 SERVICE_ID 映射到 ... 软件包名称	v2 SERVICE_ID 映射到 ... 软件包名称	v2 Maven 依赖项
ec2 `com.amazonaws.services. ec2.*`	ec2 `software.amazon.awssdk.services. ec2.*`	`<dependency> <groupId>software.amazon.awssdk</groupId> <artifactId> ec2</artifactId> </dependency>`
自动扩缩 `com.amazonaws.services. autoscaling .*`	自动扩缩 `software.amazon.awssdk.services. autoscaling .*`	`<dependency> <groupId>software.amazon.awssdk</groupId> <artifactId> autoscaling </artifactId> </dependency>`
cloudformation:* `com.amazonaws.services. cloudformation .*`	cloudformation:* `software.amazon.awssdk. cloudformation .*`	`<dependency> <groupId>software.amazon.awssdk</groupId> <artifactId> cloudformation </artifactId> </dependency>`
identitymanagement* `com.amazonaws.services. identitymanagement .*`	iam/ `software.amazon.awssdk. iam.*`	`<dependency> <groupId>software.amazon.awssdk</groupId> <artifactId> iam</artifactId> </dependency>`

* `identitymanagement` 到 `iam` 映射是一个例外，不同版本的 SERVICE_ID 有所不同。如果 Maven 或 Gradle 无法解析 v2 依赖项，请参阅the section called “软件包名称到 artifactId 的映射”以了解例外情况。

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 文件中的四个依赖项块。因为我们在步骤 1 中指定了 BOM，我们不需要添加版本。添加导入后，我们的 POM 文件具有以下依赖项元素。

```
  ...
<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 `com.amazonaws.services.ec2.AmazonEC2Client` 的 EC2 服务客户端。
- 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。我们一次只处理一个类。

a. 替换导入语句或类名称

我们看到 `describeRunningInstances` 方法的第一个参数是 v1 `AmazonEC2Client` 实例。请执行以下操作之一：

- 将 `com.amazonaws.services.ec2.AmazonEC2Client` 的导入替换为 `software.amazon.awssdk.services.ec2.Ec2Client`，然后将 `AmazonEC2Client` 更改为 `Ec2Client`。
- 将参数类型更改为 `Ec2Client`，然后让 IDE 为我们提示正确的导入。由于客户端名称不同（`AmazonEC2Client` 和 `Ec2Client`），我们的 IDE 会提示我们导入 v2 类。如果两个版本中的类名称相同，则此方法不起作用。

b. 将 v1 模型类替换为 v2 等效项

在对 v2 `Ec2Client` 进行更改之后，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);
```

编译错误是由于使用 v1 的 `DescribeInstancesRequest` 实例作为 v2 `Ec2Client` `describeInstances` 方法的参数所致。要修复此问题，请使用以下替换或导入语句。

替换	替换为
`import com.amazonaws.services.ec2.model.DescribeInstancesRequest`	`import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest`

c. 将 v1 构造函数更改为 v2 生成器。

因为 v2 类中没有构造函数，我们仍然会看到编译错误。要修复此问题，进行以下更改。

更改	到

`final DescribeInstancesRequest request = new DescribeInstancesRequest()`
`.withInstanceIds(instanceIdsCopy);`
	`final DescribeInstancesRequest request = DescribeInstancesRequest.builder()`
	`.instanceIds(instanceIdsCopy)`
	`.build();`

d. 将 v1 **`*Result`** 响应对象替换为 v2 **`*Response`** 等效项

v1 和 v2 之间一个始终一致的区别是：v2 中的所有响应对象都以 `*Response` 而不是以 `*Result` 结尾。将 v1 `DescribeInstancesResult` 导入替换为 v2 导入 `DescribeInstancesResponse`。

d. 进行 API 更改

以下语句需要进行一些更改。

```
request.setNextToken(result.getNextToken());
```

在 v2 中，setter 方法不使用 `set` `prefix`。在适用于 Java 的 SDK 2.x 中，getter 方法也不再使用 `get` 前缀

模型类（例如 `request` 实例）在 v2 中是不可变的，因此我们需要使用生成器创建一个新的 `DescribeInstancesRequest`。

在 v2 中，该语句变成了下面这样。

```
request = DescribeInstancesRequest.builder()
        .nextToken(result.nextToken())
        .build();
```

d. 重复直到方法能够使用 v2 类成功编译

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 参考和主要变更。

在我们迁移这一个方法之后，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;
    }
...
```

由于我们正在对一个包含八个方法的 Java 文件中的单个方法进行迁移，因此在处理该文件期间会同时存在 v1 和 v2 版本的导入语句。我们在执行这些步骤时添加了最后六个导入语句。

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 依赖项，从文件中移除 v1 导入

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 `dependencyManagement` 部分移除 v1 BOM（如果已使用）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代码段，我们可以选择使用 v2 分页器，让 SDK 管理基于令牌的请求，以获得更多数据。

我们可以将整个 do 子句替换为以下内容。

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

软件包名称到 Maven artifactId 的映射

当您将 Maven 或 Gradle 项目从适用于 Java 的 SDK v1 迁移到 v2 时，您需要弄清楚要将哪些依赖项添加到构建文件中。the section called “分步说明”（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项（如 artifactId）。

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 中使用的常见命名约定

下表显示了 SDK 为给定 SERVICE_ID 使用的常见命名约定。SERVICE_ID 是 AWS 服务 的唯一标识符。例如，Amazon S3 服务的 SERVICE_ID 是 `s3`，Amazon Cognito Identity 的 SERVICE_ID 是 `cognitoidentity`。

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.SERVICE_ID	aws-java-sdk-SERVICE_ID	SERVICE_ID	software.amazon.awssdk.services.SERVICE_ID
Amazon Cognito Identity 的示例（SERVICE_ID：***`cognitoidentity`*** ）			

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.cognitoidentity	aws-java-sdk-cognitoidentity	cognitoidentity	software.amazon.awssdk.services.cognitoidentity

SERVICE_ID 差异

在 v1 中

在某些情况下，同一服务的 SERVICE_ID 在软件包名称和 artifactId 中有所不同。例如，下表的 CloudWatch 指标行显示，软件包名称中的 SERVICE_ID 是 `metrics`，而 artifactId 的 SERVICE_ID 是 `cloudwatchmetrics`。

在 v2 中

软件包名称和 artifactId 中使用的 SERVICE_ID 没有差异。

v1 和 v2 之间

对于大多数服务，v2 中的 SERVICE_ID 与 v1 在软件包名称和 artifactId 中的 SERVICE_ID 相同。这方面的一个例子是 `cognitoedentity` SERVICE_ID，如上表所示。但是，某些 Service_ID 在不同 SDK 版本之间有所不同，如下表所示。

任一个 v1 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 `com.amazonaws.services` 开头，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，如第一行所示。	所有软件包名称都以 `software.amazon.awssdk` 开头，如第一行所示。
API Gateway	com.amazonaws.services.apigateway	<artifactId>aws-java-sdk-api-	<artifactId>apigateway</artifactId>	software.amazon.aw

服务名称	v1 软件包名称	v1 artifactId	v2 artifactId	v2 软件包名称
		gateway</artifactId>		ssdk.services.apigateway
App Registry	appregistry	appregistry	servicecatalogappregistry	servicecatalogappregistry
Application Discovery	applicationdiscovery	discovery	applicationdiscovery	applicationdiscovery
Augmented AI Runtime	augmentedairuntime	augmentedairuntime	sagemakera2iruntime	sagemakera2iruntime
Certificate Manager	certificatemanager	acm	acm	acm
CloudControl API	cloudcontrolapi	cloudcontrolapi	cloudcontrol	cloudcontrol
CloudSearch	cloudsearchv2	cloudsearch	cloudsearch	cloudsearch
CloudSearch Domain	cloudsearchdomain	cloudsearch	cloudsearchdomain	cloudsearchdomain
CloudWatch Events	cloudwatchevents	events	cloudwatchevents	cloudwatchevents
CloudWatch Evidently	cloudwatchevidently	cloudwatchevidently	evidently	evidently
CloudWatch Logs	logs	logs	cloudwatchlogs	cloudwatchlogs
CloudWatch Metrics	metrics	cloudwatchmetrics	cloudwatch	cloudwatch
CloudWatch Rum	cloudwatchrum	cloudwatchrum	rum	rum

服务名称	v1 软件包名称	v1 artifactId	v2 artifactId	v2 软件包名称
Cognito Identity Provider	cognitoidp	cognitoidp	cognitoid entityprovider	cognitoid entityprovider
Connect Campaign	connectca mpaign	connectca mpaign	connectca mpaigns	connectca mpaigns
Connect Wisdom	connectwisdom	connectwisdom	wisdom	wisdom
Database Migration Service	databasem igrationservice	dms	databasem igration	databasem igration
DataZone	datazone	datazoneexternal	datazone	datazone
DynamoDB	dynamodbv2	dynamodb	dynamodb	dynamodb
Elastic File System	elasticfilesystem	efs	efs	efs
Elastic Map Reduce	elasticma preduce/	emr	emr	emr
Glue DataBrew	gluedatabrew	gluedatabrew	databrew	databrew
IAM Roles Anywhere	iamrolesa nywhere	iamrolesa nywhere	rolesanywhere	rolesanywhere
身份管理	identitym anagement	IAM	IAM	IAM
IoT Data	iotdata	iot	iotdataplane	iotdataplane
Kinesis Analytics	kinesisanalytics	kinesis –	kinesisanalytics	kinesisanalytics
Kinesis Firehose	kinesisfirehose	kinesis –	Firehose	Firehose

服务名称	v1 软件包名称	v1 artifactId	v2 artifactId	v2 软件包名称
Kinesis Video Signaling Channels	kinesisvideosignalingchannels	kinesisvideosignalingchannels	kinesisvideosignaling	kinesisvideosignaling
Lex	lexruntime	Lex	lexruntime	lexruntime
Lookout For Vision	lookoutforvision	lookoutforvision	lookoutvision	lookoutvision
Mainframe Modernization	mainframemodernization	mainframemodernization	m2	m2
Marketplace Metering	marketplacemetering	marketplacemeteringservice	marketplacemetering	marketplacemetering
Managed Grafana	managedgrafana	managedgrafana	grafana	grafana
Mechanical Turk	mturk	mechanicalturkrequester	mturk	mturk
Migration Hub 策略建议	migrationhubstrategyrecommendations	migrationhubstrategyrecommendations	migrationhubstrategy	migrationhubstrategy
Nimble Studio	nimblestudio	nimblestudio	nimble	nimble
Private 5G	private5g	private5g	privatenetworks	privatenetworks
Prometheus	prometheus	prometheus	amp	amp
Recycle Bin	recyclebin	recyclebin	rbin	rbin
Redshift 数据 API	redshiftdataapi	redshiftdataapi	redshiftdata	redshiftdata

服务名称	v1 软件包名称	v1 artifactId	v2 artifactId	v2 软件包名称
Route 53	route53domains	route53	route53domains	route53domains
Sage Maker Edge Manager	sagemaker edgemanager	sagemaker edgemanager	sagemakeredge	sagemakeredge
Security Token	securitytoken	sts	sts	sts
Server Migration	servermigration	servermigration	sms	sms
Simple Email	simpleemail	ses	ses	ses
Simple Email V2	simpleemailv2	sesv2	sesv2	sesv2
Simple Systems Management	simplesys temsmanag ement	ssm	ssm	ssm
Simple Workflow	simpleworkflow	simpleworkflow	swf	swf
Step Functions	stepfunctions	stepfunctions	sfn	sfn

AWS SDK for Java 1.x 和 2.x 之间的差异

本部分介绍在将应用程序从使用AWS SDK for Java 版本 1.x 转换为使用版本 2.x 时需要注意的主要更改。

软件包名称更改

从 SDK for Java 1.x 到 SDK for Java 2.x 的一个明显变化是软件包名称的更改。软件包名称在 SDK 2.x 中以 `software.amazon.awssdk` 开头，而 SDK 1.x 则使用 `com.amazonaws`。

这些名称区分了 SDK 1.x 与 SDK 2.x 的 Maven 构件。SDK 2.x 的 Maven 构件使用 `software.amazon.awssdk` groupId，而 SDK 1.x 使用 `com.amazonaws` groupId。

有些时候，您的代码会要求项目具有 `com.amazonaws` 依赖项，而该项目原本只使用 SDK 2.x 构件。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 com.amazonaws 开头，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 方法来管理依赖项。要将版本 2.x 组件添加到您的项目，只需使用 SDK 上的依赖项来更新 pom.xml 文件。

Example

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

<dependencies>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>dynamodb</artifactId>
    </dependency>
</dependencies>
```

在将项目迁移到版本 2.x 时，您也可以并行使用版本 1.x 和 2.x。

不可变 POJO

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();
```

Setter 和 getter 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 `get` 前缀。

Example在 1.x 中使用 setter 方法

```
AmazonDynamoDB client = AmazonDynamoDBClientBuilder.standard()
          .withRegion("us-east-1")
          .build();
```

Example在 2.x 中使用 setter 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在 v2 中以 `Response` 结尾，而不是以 v1 使用的 `Result` 结尾。

Example在 v1 中代表响应的类名

```
CreateApiKeyResult
AllocateAddressResult
```

Example在 v2 中代表响应的类名

```
CreateApiKeyResponse
AllocateAddressResponse
```

库和实用工具的迁移状态

适用于 Java 的 SDK 库和实用工具

下表列出了适用于 Java 的 SDK 库和实用工具的迁移状态。

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
CloudFrontUrlSigner、CloudFrontCookieSigner	CloudFrontUtilities	2.18.33
TransferManager	S3TransferManager	2.19.0
EC2 元数据客户端	EC2 元数据客户端	2.19.29

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
S3 URI 解析器	S3 URI 解析器	2.20.41
IAM policy 生成器	IAM 策略生成器	2.20.126
S3 事件通知	S3 事件通知	2.25.11
Amazon SQS 客户端缓存	适用于 Amazon SQS 的自动请求批处理 API	2.28.0
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可与适用于 Java 的 SDK 2.x 配合使用的库。

适用于 Java 的 SDK 2.x 中使用的名称	最低版本
Amazon S3 加密客户端	3.0.0[1]
适用于 DynamoDB 的 AWS 数据库加密 SDK	3.0.0[2]

[1]Amazon S3 加密客户端

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]适用于 DynamoDB 的 AWS 数据库加密 SDK

适用于 DynamoDB 的 AWS 数据库加密 SDK 现可与 V2 配合使用，可通过使用以下 Maven 依赖项获得。

```
<dependency>
```

```
    <groupId>software.amazon.cryptography</groupId>
    <artifactId>aws-database-encryption-sdk-dynamodb</artifactId>
    <version>3.x</version>
</dependency>
```

《AWS 数据库加密 SDK 开发人员指南》（名为适用于 Java 的 Amazon DynamoDB Encryption Client）和 GitHub 中提供了与 Java SDK v1 配合使用的 DynamoDB 加密库的信息。

有关与 Java SDK V2 兼容的 DynamoDB 加密库的更多信息，请参阅《AWS 数据库加密 SDK 开发人员指南》和 GitHub 源。

《AWS 数据库加密 SDK 开发人员指南》中提供了有关加密库的迁移信息。

库和实用工具的迁移详细信息

- Transfer Manager
- EC2 元数据实用程序
- CloudFront 预签名
- S3 URI 解析
- DynamoDB 映射/文档 API
- IAM 策略生成器
- S3 事件通知
- SDK 指标发布（1.x 文档、2.x 文档）

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();
```

客户端类名

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
AWSACMPCAAsyncClient
```

Example 2.x 中的类名

```
DynamoDbClient
AcmAsyncClient
```

客户端类名更改

1.x 客户端	2.x 客户端
com.amazonaws.services.acmpca.AWSACMPCAAsyncClient	software.amazon.awssdk.services.acm.AcmAsyncClient
com.amazonaws.services.acmpca.AWSACMPCAClient	software.amazon.awssdk.services.acm.AcmClient
com.amazonaws.services.alexaforbusiness.AmazonAlexaForBusinessAsyncClient	software.amazon.awssdk.services.alexaforbusiness.AlexaForBusinessAsyncClient
com.amazonaws.services.alexaforbusiness.AmazonAlexaForBusinessClient	software.amazon.awssdk.services.alexaforbusiness.AlexaForBusinessClient
com.amazonaws.services.apigateway.AmazonApiGatewayAsyncClient	software.amazon.awssdk.services.apigateway.ApiGatewayAsyncClient

1.x 客户端	2.x 客户端
com.amazonaws.services.apigateway.AmazonApiGatewayClient	software.amazon.awssdk.services.apigateway.ApiGatewayClient
com.amazonaws.services.applicationautoscaling.AWSApplicationAutoScalingAsyncClient	software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingAsyncClient
com.amazonaws.services.applicationautoscaling.AWSApplicationAutoScalingClient	software.amazon.awssdk.services.applicationautoscaling.ApplicationAutoScalingClient
com.amazonaws.services.applicationdiscovery.AWSApplicationDiscoveryAsyncClient	software.amazon.awssdk.services.applicationdiscovery.ApplicationDiscoveryAsyncClient
com.amazonaws.services.applicationdiscovery.AWSApplicationDiscoveryClient	software.amazon.awssdk.services.applicationdiscovery.ApplicationDiscoveryClient
com.amazonaws.services.appstream.AmazonAppStreamAsyncClient	software.amazon.awssdk.services.appstream.AppStreamAsyncClient
com.amazonaws.services.appstream.AmazonAppStreamClient	software.amazon.awssdk.services.appstream.AppStreamClient
com.amazonaws.services.appsync.AWSAppSyncAsyncClient	software.amazon.awssdk.services.appsync.AppSyncAsyncClient
com.amazonaws.services.appsync.AWSAppSyncClient	software.amazon.awssdk.services.appsync.AppSyncClient
com.amazonaws.services.athena.AmazonAthenaAsyncClient	software.amazon.awssdk.services.athena.AthenaAsyncClient

1.x 客户端	2.x 客户端
com.amazonaws.services.athena.AmazonAthenaClient	software.amazon.awssdk.services.athena.AthenaClient
com.amazonaws.services.autoscaling.AmazonAutoScalingAsyncClient	software.amazon.awssdk.services.autoscaling.AutoScalingAsyncClient
com.amazonaws.services.autoscaling.AmazonAutoScalingClient	software.amazon.awssdk.services.autoscaling.AutoScalingClient
com.amazonaws.services.autoscalingplans.AWSAutoScalingPlansAsyncClient	software.amazon.awssdk.services.autoscalingplans.AutoScalingPlansAsyncClient
com.amazonaws.services.autoscalingplans.AWSAutoScalingPlansClient	software.amazon.awssdk.services.autoscalingplans.AutoScalingPlansClient
com.amazonaws.services.batch.AWSBatchAsyncClient	software.amazon.awssdk.services.batch.BatchAsyncClient
com.amazonaws.services.batch.AWSBatchClient	software.amazon.awssdk.services.batch.BatchClient
com.amazonaws.services.budgets.AWSBudgetsAsyncClient	software.amazon.awssdk.services.budgets.BudgetsAsyncClient
com.amazonaws.services.budgets.AWSBudgetsClient	software.amazon.awssdk.services.budgets.BudgetsClient
com.amazonaws.services.certificatemanager.AWSCertificateManagerAsyncClient	software.amazon.awssdk.services.acm.AcmAsyncClient

1.x 客户端	2.x 客户端
com.amazonaws.services.certificatemanager.AWSCertificateManagerClient	software.amazon.awssdk.services.acm.AcmClient
com.amazonaws.services.cloud9.AWSCloud9AsyncClient	software.amazon.awssdk.services.cloud9.Cloud9AsyncClient
com.amazonaws.services.cloud9.AWSCloud9Client	software.amazon.awssdk.services.cloud9.Cloud9Client
com.amazonaws.services.clouddirectory.AmazonCloudDirectoryAsyncClient	software.amazon.awssdk.services.clouddirectory.CloudDirectoryAsyncClient
com.amazonaws.services.clouddirectory.AmazonCloudDirectoryClient	software.amazon.awssdk.services.clouddirectory.CloudDirectoryClient
com.amazonaws.services.cloudformation.AmazonCloudFormationAsyncClient	software.amazon.awssdk.services.cloudformation.CloudFormationAsyncClient
com.amazonaws.services.cloudformation.AmazonCloudFormationClient	software.amazon.awssdk.services.cloudformation.CloudFormationClient
com.amazonaws.services.cloudfront.AmazonCloudFrontAsyncClient	software.amazon.awssdk.services.cloudfront.CloudFrontAsyncClient
com.amazonaws.services.cloudfront.AmazonCloudFrontClient	software.amazon.awssdk.services.cloudfront.CloudFrontClient
com.amazonaws.services.cloudhsm.AWSCloudHSMAsyncClient	software.amazon.awssdk.services.cloudhsm.CloudHsmAsyncClient

1.x 客户端	2.x 客户端
com.amazonaws.services.cloudhsm.AWSCloudHSMClient	software.amazon.awssdk.services.cloudhsm.CloudHsmClient
com.amazonaws.services.cloudhsmv2.AWSCloudHSMV2AsyncClient	software.amazon.awssdk.services.cloudhsmv2.CloudHsmV2AsyncClient
com.amazonaws.services.cloudhsmv2.AWSCloudHSMV2Client	software.amazon.awssdk.services.cloudhsmv2.CloudHsmV2Client
com.amazonaws.services.cloudsearchdomain.AmazonCloudSearchDomainAsyncClient	software.amazon.awssdk.services.cloudsearchdomain.CloudSearchDomainAsyncClient
com.amazonaws.services.cloudsearchdomain.AmazonCloudSearchDomainClient	software.amazon.awssdk.services.cloudsearchdomain.CloudSearchDomainClient
com.amazonaws.services.cloudsearchv2.AmazonCloudSearchAsyncClient	software.amazon.awssdk.services.cloudsearch.CloudSearchAsyncClient
com.amazonaws.services.cloudsearchv2.AmazonCloudSearchClient	software.amazon.awssdk.services.cloudsearch.CloudSearchClient
com.amazonaws.services.cloudtrail.AWSCloudTrailAsyncClient	software.amazon.awssdk.services.cloudtrail.CloudTrailAsyncClient
com.amazonaws.services.cloudtrail.AWSCloudTrailClient	software.amazon.awssdk.services.cloudtrail.CloudTrailClient

1.x 客户端	2.x 客户端
com.amazonaws.services.cloudwatch.AmazonCloudWatchAsyncClient	software.amazon.awssdk.services.cloudwatch.CloudWatchAsyncClient
com.amazonaws.services.cloudwatch.AmazonCloudWatchClient	software.amazon.awssdk.services.cloudwatch.CloudWatchClient
com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEventsAsyncClient	software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsAsyncClient
com.amazonaws.services.cloudwatchevents.AmazonCloudWatchEventsClient	software.amazon.awssdk.services.cloudwatchevents.CloudWatchEventsClient
com.amazonaws.services.codebuild.AWSCodeBuildAsyncClient	software.amazon.awssdk.services.codebuild.CodeBuildAsyncClient
com.amazonaws.services.codebuild.AWSCodeBuildClient	software.amazon.awssdk.services.codebuild.CodeBuildClient
com.amazonaws.services.codecommit.AWSCodeCommitAsyncClient	software.amazon.awssdk.services.codecommit.CodeCommitAsyncClient
com.amazonaws.services.codecommit.AWSCodeCommitClient	software.amazon.awssdk.services.codecommit.CodeCommitClient
com.amazonaws.services.codedeploy.AmazonCodeDeployAsyncClient	software.amazon.awssdk.services.codedeploy.CodeDeployAsyncClient

1.x 客户端	2.x 客户端
com.amazonaws.services.codedeploy.AmazonCodeDeployClient	software.amazon.awssdk.services.codedeploy.CodeDeployClient
com.amazonaws.services.codepipeline.AWSCodePipelineAsyncClient	software.amazon.awssdk.services.codepipeline.CodePipelineAsyncClient
com.amazonaws.services.codepipeline.AWSCodePipelineClient	software.amazon.awssdk.services.codepipeline.CodePipelineClient
com.amazonaws.services.codestar.AWSCodeStarAsyncClient	software.amazon.awssdk.services.codestar.CodeStarAsyncClient
com.amazonaws.services.codestar.AWSCodeStarClient	software.amazon.awssdk.services.codestar.CodeStarClient
com.amazonaws.services.cognitoidentity.AmazonCognitoIdentityAsyncClient	software.amazon.awssdk.services.cognitoidentity.CognitoIdentityAsyncClient
com.amazonaws.services.cognitoidentity.AmazonCognitoIdentityClient	software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient
com.amazonaws.services.cognitoidp.AWSCognitoIdentityProviderAsyncClient	software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderAsyncClient
com.amazonaws.services.cognitoidp.AWSCognitoIdentityProviderClient	software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient

1.x 客户端	2.x 客户端
com.amazonaws.services.cognitosync.AmazonCognitoSyncAsyncClient	software.amazon.awssdk.services.cognitosync.CognitoSyncAsyncClient
com.amazonaws.services.cognitosync.AmazonCognitoSyncClient	software.amazon.awssdk.services.cognitosync.CognitoSyncClient
com.amazonaws.services.comprehend.AmazonComprehendAsyncClient	software.amazon.awssdk.services.comprehend.ComprehendAsyncClient
com.amazonaws.services.comprehend.AmazonComprehendClient	software.amazon.awssdk.services.comprehend.ComprehendClient
com.amazonaws.services.config.AmazonConfigAsyncClient	software.amazon.awssdk.services.config.ConfigAsyncClient
com.amazonaws.services.config.AmazonConfigClient	software.amazon.awssdk.services.config.ConfigClient
com.amazonaws.services.connect.AmazonConnectAsyncClient	software.amazon.awssdk.services.connect.ConnectAsyncClient
com.amazonaws.services.connect.AmazonConnectClient	software.amazon.awssdk.services.connect.ConnectClient
com.amazonaws.services.costandusagereport.AWSCostAndUsageReportAsyncClient	software.amazon.awssdk.services.costandusagereport.CostAndUsageReportAsyncClient
com.amazonaws.services.costandusagereport.AWSCostAndUsageReportClient	software.amazon.awssdk.services.costandusagereport.CostAndUsageReportClient

1.x 客户端	2.x 客户端
com.amazonaws.services.costexplorer.AWSCostExplorerAsyncClient	software.amazon.awssdk.services.costexplorer.CostExplorerAsyncClient
com.amazonaws.services.costexplorer.AWSCostExplorerClient	software.amazon.awssdk.services.costexplorer.CostExplorerClient
com.amazonaws.services.databasemigrationservice.AWSDatabaseMigrationServiceAsyncClient	software.amazon.awssdk.services.databasemigration.DatabaseMigrationAsyncClient
com.amazonaws.services.databasemigrationservice.AWSDatabaseMigrationServiceClient	software.amazon.awssdk.services.databasemigration.DatabaseMigrationClient
com.amazonaws.services.datapipeline.DataPipelineAsyncClient	software.amazon.awssdk.services.datapipeline.DataPipelineAsyncClient
com.amazonaws.services.datapipeline.DataPipelineClient	software.amazon.awssdk.services.datapipeline.DataPipelineAsyncClient
com.amazonaws.services.dax.AmazonDaxAsyncClient	software.amazon.awssdk.services.dax.DaxAsyncClient
com.amazonaws.services.dax.AmazonDaxClient	software.amazon.awssdk.services.dax.DaxClient
com.amazonaws.services.devicefarm.AWSDeviceFarmAsyncClient	software.amazon.awssdk.services.devicefarm.DeviceFarmAsyncClient

1.x 客户端	2.x 客户端
com.amazonaws.services.devicefarm.AWSDeviceFarmClient	software.amazon.awssdk.services.devicefarm.DeviceFarmClient
com.amazonaws.services.directconnect.AmazonDirectConnectAsyncClient	software.amazon.awssdk.services.directconnect.DirectConnectAsyncClient
com.amazonaws.services.directconnect.AmazonDirectConnectClient	software.amazon.awssdk.services.directconnect.DirectConnectClient
com.amazonaws.services.directory.AWSDirectoryServiceAsyncClient	software.amazon.awssdk.services.directory.DirectoryAsyncClient
com.amazonaws.services.directory.AWSDirectoryServiceClient	software.amazon.awssdk.services.directory.DirectoryClient
com.amazonaws.services.dlm.AmazonDLMAsyncClient	software.amazon.awssdk.services.dlm.DlmAsyncClient
com.amazonaws.services.dlm.AmazonDLMClient	software.amazon.awssdk.services.dlm.DlmClient
com.amazonaws.services.dynamodbv2.AmazonDynamoDBAsyncClient	software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient
com.amazonaws.services.dynamodbv2.AmazonDynamoDBClient	software.amazon.awssdk.services.dynamodb.DynamoDbClient
com.amazonaws.services.dynamodbv2.AmazonDynamoDBStreamsAsyncClient	software.amazon.awssdk.services.dynamodb.streams.DynamoDbStreamsAsyncClient

1.x 客户端	2.x 客户端
com.amazonaws.services.dynamodbv2.AmazonDynamoDBStreamsClient	software.amazon.awssdk.services.dynamodb.streams.DynamoDbStreamsClient
com.amazonaws.services.ec2.AmazonEC2AsyncClient	software.amazon.awssdk.services.ec2.Ec2AsyncClient
com.amazonaws.services.ec2.AmazonEC2Client	software.amazon.awssdk.services.ec2.Ec2Client
com.amazonaws.services.ecr.AmazonECRAsyncClient	software.amazon.awssdk.services.ecr.EcrAsyncClient
com.amazonaws.services.ecr.AmazonECRClient	software.amazon.awssdk.services.ecr.EcrClient
com.amazonaws.services.ecs.AmazonECSAsyncClient	software.amazon.awssdk.services.ecs.EcsAsyncClient
com.amazonaws.services.ecs.AmazonECSClient	software.amazon.awssdk.services.ecs.EcsClient
com.amazonaws.services.eks.AmazonEKSAsyncClient	software.amazon.awssdk.services.eks.EksAsyncClient
com.amazonaws.services.eks.AmazonEKSClient	software.amazon.awssdk.services.eks.EksClient
com.amazonaws.services.elasticache.AmazonElastiCacheAsyncClient	software.amazon.awssdk.services.elasticache.ElastiCacheAsyncClient
com.amazonaws.services.elasticache.AmazonElastiCacheClient	software.amazon.awssdk.services.elasticache.ElastiCacheClient

1.x 客户端	2.x 客户端
com.amazonaws.services.elasticbeanstalk.AWSElasticBeanstalkAsyncClient	software.amazon.awssdk.services.elasticbeanstalk.ElasticBeanstalkAsyncClient
com.amazonaws.services.elasticbeanstalk.AWSElasticBeanstalkClient	software.amazon.awssdk.services.elasticbeanstalk.ElasticBeanstalkClient
com.amazonaws.services.elasticfilesystem.AmazonElasticFileSystemAsyncClient	software.amazon.awssdk.services.efs.EfsAsyncClient
com.amazonaws.services.elasticfilesystem.AmazonElasticFileSystemClient	software.amazon.awssdk.services.efs.EfsClient
com.amazonaws.services.elasticloadbalancing.AmazonElasticLoadBalancingAsyncClient	software.amazon.awssdk.services.elasticloadbalancing.ElasticLoadBalancingAsyncClient
com.amazonaws.services.elasticloadbalancing.AmazonElasticLoadBalancingClient	software.amazon.awssdk.services.elasticloadbalancing.ElasticLoadBalancingClient
com.amazonaws.services.elasticloadbalancingv2.AmazonElasticLoadBalancingAsyncClient	software.amazon.awssdk.services.elasticloadbalancingv2.ElasticLoadBalancingV2AsyncClient
com.amazonaws.services.elasticloadbalancingv2.AmazonElasticLoadBalancingClient	software.amazon.awssdk.services.elasticloadbalancingv2.ElasticLoadBalancingV2Client
com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduceAsyncClient	software.amazon.awssdk.services.emr.EmrAsyncClient

1.x 客户端	2.x 客户端
com.amazonaws.services.elasticmapreduce.AmazonElasticMapReduceClient	software.amazon.awssdk.services.emr.EmrClient
com.amazonaws.services.elasticsearch.AWSElasticsearchAsyncClient	software.amazon.awssdk.services.elasticsearch.ElasticsearchAsyncClient
com.amazonaws.services.elasticsearch.AWSElasticsearchClient	software.amazon.awssdk.services.elasticsearch.ElasticsearchClient
com.amazonaws.services.elastictranscoder.AmazonElasticTranscoderAsyncClient	software.amazon.awssdk.services.elastictranscoder.ElasticTranscoderAsyncClient
com.amazonaws.services.elastictranscoder.AmazonElasticTranscoderClient	software.amazon.awssdk.services.elastictranscoder.ElasticTranscoderClient
com.amazonaws.services.fms.AWSFMSAsyncClient	software.amazon.awssdk.services.fms.FmsAsyncClient
com.amazonaws.services.fms.AWSFMSClient	software.amazon.awssdk.services.fms.FmsClient
com.amazonaws.services.gamelift.AmazonGameLiftAsyncClient	software.amazon.awssdk.services.gamelift.GameLiftAsyncClient
com.amazonaws.services.gamelift.AmazonGameLiftClient	software.amazon.awssdk.services.gamelift.GameLiftClient
com.amazonaws.services.glacier.AmazonGlacierAsyncClient	software.amazon.awssdk.services.glacier.GlacierAsyncClient

1.x 客户端	2.x 客户端
com.amazonaws.services.glacier.AmazonGlacierClient	software.amazon.awssdk.services.glacier.GlacierClient
com.amazonaws.services.glue.AWSGlueAsyncClient	software.amazon.awssdk.services.glue.GlueAsyncClient
com.amazonaws.services.glue.AWSGlueClient	software.amazon.awssdk.services.glue.GlueClient
com.amazonaws.services.greengrass.AWSGreengrassAsyncClient	software.amazon.awssdk.services.greengrass.GreengrassAsyncClient
com.amazonaws.services.greengrass.AWSGreengrassClient	software.amazon.awssdk.services.greengrass.GreengrassClient
com.amazonaws.services.guardduty.AmazonGuardDutyAsyncClient	software.amazon.awssdk.services.guardduty.GuardDutyAsyncClient
com.amazonaws.services.guardduty.AmazonGuardDutyClient	software.amazon.awssdk.services.guardduty.GuardDutyClient
com.amazonaws.services.health.AWSHealthAsyncClient	software.amazon.awssdk.services.health.HealthAsyncClient
com.amazonaws.services.health.AWSHealthClient	software.amazon.awssdk.services.health.HealthClient
com.amazonaws.services.identitymanagement.AmazonIdentityManagementAsyncClient	software.amazon.awssdk.services.iam.IamAsyncClient
com.amazonaws.services.identitymanagement.AmazonIdentityManagementClient	software.amazon.awssdk.services.iam.IamClient

1.x 客户端	2.x 客户端
com.amazonaws.services.importexport.AmazonImportExportAsyncClient	###
com.amazonaws.services.importexport.AmazonImportExportClient	###
com.amazonaws.services.inspector.AmazonInspectorAsyncClient	software.amazon.awssdk.services.inspector.InspectorAsyncClient
com.amazonaws.services.inspector.AmazonInspectorClient	software.amazon.awssdk.services.inspector.InspectorClient
com.amazonaws.services.iot.AWSIotAsyncClient	software.amazon.awssdk.services.iot.IotAsyncClient
com.amazonaws.services.iot.AWSIotClient	software.amazon.awssdk.services.iot.IotClient
com.amazonaws.services.iot1clickdevices.AWSIoT1ClickDevicesAsyncClient	software.amazon.awssdk.services.iot1clickdevices.Iot1ClickDevicesAsyncClient
com.amazonaws.services.iot1clickdevices.AWSIoT1ClickDevicesClient	software.amazon.awssdk.services.iot1clickdevices.Iot1ClickDevicesClient
com.amazonaws.services.iot1clickprojects.AWSIoT1ClickProjectsAsyncClient	software.amazon.awssdk.services.iot1clickprojects.Iot1ClickProjectsAsyncClient
com.amazonaws.services.iot1clickprojects.AWSIoT1ClickProjectsClient	software.amazon.awssdk.services.iot1clickprojects.Iot1ClickProjectsClient

1.x 客户端	2.x 客户端
com.amazonaws.services.iotanalytics.AWSIoTAnalyticsAsyncClient	software.amazon.awssdk.services.iotanalytics.IotAnalyticsAsyncClient
com.amazonaws.services.iotanalytics.AWSIoTAnalyticsClient	software.amazon.awssdk.services.iotanalytics.IotAnalyticsClient
com.amazonaws.services.iotdata.AWSIotDataAsyncClient	software.amazon.awssdk.services.iotdata.IotDataAsyncClient
com.amazonaws.services.iotdata.AWSIotDataClient	software.amazon.awssdk.services.iotdata.IotDataClient
com.amazonaws.services.iotjobsdataplane.AWSIoTJobsDataPlaneAsyncClient	software.amazon.awssdk.services.iotdataplane.IotDataPlaneAsyncClient
com.amazonaws.services.iotjobsdataplane.AWSIoTJobsDataPlaneClient	software.amazon.awssdk.services.iotdataplane.IotDataPlaneClient
com.amazonaws.services.kinesis.AmazonKinesisAsyncClient	software.amazon.awssdk.services.kinesis.KinesisAsyncClient
com.amazonaws.services.kinesis.AmazonKinesisClient	software.amazon.awssdk.services.kinesis.KinesisClient
com.amazonaws.services.kinesisanalytics.AmazonKinesisAnalyticsAsyncClient	software.amazon.awssdk.services.kinesisanalytics.KinesisAnalyticsAsyncClient
com.amazonaws.services.kinesisanalytics.AmazonKinesisAnalyticsClient	software.amazon.awssdk.services.kinesisanalytics.KinesisAnalyticsClient

1.x 客户端	2.x 客户端
com.amazonaws.services.kinesisfirehose.AmazonKinesisFirehoseAsyncClient	software.amazon.awssdk.services.firehose.FirehoseAsyncClient
com.amazonaws.services.kinesisfirehose.AmazonKinesisFirehoseClient	software.amazon.awssdk.services.firehose.FirehoseClient
com.amazonaws.services.kinesisvideo.AmazonKinesisVideoArchivedMediaAsyncClient	software.amazon.awssdk.services.kinesisvideoarchivedmedia.KinesisVideoArchivedMediaAsyncClient
com.amazonaws.services.kinesisvideo.AmazonKinesisVideoArchivedMediaClient	software.amazon.awssdk.services.kinesisvideoarchivedmedia.KinesisVideoArchivedMediaClient
com.amazonaws.services.kinesisvideo.AmazonKinesisVideoAsyncClient	software.amazon.awssdk.services.kinesisvideo.KinesisVideoAsyncClient
com.amazonaws.services.kinesisvideo.AmazonKinesisVideoClient	software.amazon.awssdk.services.kinesisvideo.KinesisVideoClient
com.amazonaws.services.kinesisvideo.AmazonKinesisVideoMediaAsyncClient	software.amazon.awssdk.services.kinesisvideomedia.KinesisVideoMediaAsyncClient
com.amazonaws.services.kinesisvideo.AmazonKinesisVideoMediaClient	software.amazon.awssdk.services.kinesisvideomedia.KinesisVideoMediaClient

1.x 客户端	2.x 客户端
com.amazonaws.services.kinesisvideo.AmazonKinesisVideoPutMediaClient	不支持
com.amazonaws.services.kms.AWSKMSAsyncClient	software.amazon.awssdk.services.kms.KmsAsyncClient
com.amazonaws.services.kms.AWSKMSClient	software.amazon.awssdk.services.kms.KmsClient
com.amazonaws.services.lambda.AWSLambdaAsyncClient	software.amazon.awssdk.services.lambda.LambdaAsyncClient
com.amazonaws.services.lambda.AWSLambdaClient	software.amazon.awssdk.services.lambda.LambdaClient
com.amazonaws.services.lexmodelbuilding.AmazonLexModelBuildingAsyncClient	software.amazon.awssdk.services.lexmodelbuilding.LexModelBuildingAsyncClient
com.amazonaws.services.lexmodelbuilding.AmazonLexModelBuildingClient	software.amazon.awssdk.services.lexmodelbuilding.LexModelBuildingClient
com.amazonaws.services.lexruntime.AmazonLexRuntimeAsyncClient	software.amazon.awssdk.services.lexruntime.LexRuntimeAsyncClient
com.amazonaws.services.lexruntime.AmazonLexRuntimeClient	software.amazon.awssdk.services.lexruntime.LexRuntimeClient
com.amazonaws.services.lightsail.AmazonLightsailAsyncClient	software.amazon.awssdk.services.lightsail.LightsailAsyncClient

1.x 客户端	2.x 客户端
com.amazonaws.services.lightsail.AmazonLightsailClient	software.amazon.awssdk.services.lightsail.LightsailClient
com.amazonaws.services.logs.AWSLogsAsyncClient	software.amazon.awssdk.services.logs.LogsAsyncClient
com.amazonaws.services.logs.AWSLogsClient	software.amazon.awssdk.services.logs.LogsClient
com.amazonaws.services.machinelearning.AmazonMachineLearningAsyncClient	software.amazon.awssdk.services.machinelearning.MachineLearningAsyncClient
com.amazonaws.services.machinelearning.AmazonMachineLearningClient	software.amazon.awssdk.services.machinelearning.MachineLearningClient
com.amazonaws.services.macie.AmazonMacieAsyncClient	software.amazon.awssdk.services.macie.MacieAsyncClient
com.amazonaws.services.macie.AmazonMacieClient	software.amazon.awssdk.services.macie.MacieClient
com.amazonaws.services.marketplacecommerceanalytics.AWSMarketplaceCommerceAnalyticsAsyncClient	software.amazon.awssdk.services.marketplacecommerceanalytics.MarketplaceCommerceAnalyticsAsyncClient
com.amazonaws.services.marketplacecommerceanalytics.AWSMarketplaceCommerceAnalyticsClient	software.amazon.awssdk.services.marketplacecommerceanalytics.MarketplaceCommerceAnalyticsClient
com.amazonaws.services.marketplaceentitlement.AWSMarketplaceEntitlementAsyncClient	software.amazon.awssdk.services.marketplaceentitlement.MarketplaceEntitlementAsyncClient

1.x 客户端	2.x 客户端
com.amazonaws.services.marketplaceentitlement.AWSMarketplaceEntitlementClient	software.amazon.awssdk.services.marketplaceentitlement.MarketplaceEntitlementClient
com.amazonaws.services.marketplacemetering.AWSMarketplaceMeteringAsyncClient	software.amazon.awssdk.services.marketplacemetering.MarketplaceMeteringAsyncClient
com.amazonaws.services.marketplacemetering.AWSMarketplaceMeteringClient	software.amazon.awssdk.services.marketplacemetering.MarketplaceMeteringClient
com.amazonaws.services.mediaconvert.AWSMediaConvertAsyncClient	software.amazon.awssdk.services.mediaconvert.MediaConvertAsyncClient
com.amazonaws.services.mediaconvert.AWSMediaConvertClient	software.amazon.awssdk.services.mediaconvert.MediaConvertClient
com.amazonaws.services.medialive.AWSMediaLiveAsyncClient	software.amazon.awssdk.services.medialive.MediaLiveAsyncClient
com.amazonaws.services.medialive.AWSMediaLiveClient	software.amazon.awssdk.services.medialive.MediaLiveClient
com.amazonaws.services.mediapackage.AWSMediaPackageAsyncClient	software.amazon.awssdk.services.mediapackage.MediaPackageAsyncClient
com.amazonaws.services.mediapackage.AWSMediaPackageClient	software.amazon.awssdk.services.mediapackage.MediaPackageClient

1.x 客户端	2.x 客户端
com.amazonaws.services.mediastore.AWSMediaStoreAsyncClient	software.amazon.awssdk.services.mediastore.MediaStoreAsyncClient
com.amazonaws.services.mediastore.AWSMediaStoreClient	software.amazon.awssdk.services.mediastore.MediaStoreClient
com.amazonaws.services.mediastoredata.AWSMediaStoreDataAsyncClient	software.amazon.awssdk.services.mediastoredata.MediaStoreDataAsyncClient
com.amazonaws.services.mediastoredata.AWSMediaStoreDataClient	software.amazon.awssdk.services.mediastoredata.MediaStoreDataClient
com.amazonaws.services.mediatailor.AWSMediaTailorAsyncClient	software.amazon.awssdk.services.mediatailor.MediaTailorAsyncClient
com.amazonaws.services.mediatailor.AWSMediaTailorClient	software.amazon.awssdk.services.mediatailor.MediaTailorClient
com.amazonaws.services.migrationhub.AWSMigrationHubAsyncClient	software.amazon.awssdk.services.migrationhub.MigrationHubAsyncClient
com.amazonaws.services.migrationhub.AWSMigrationHubClient	software.amazon.awssdk.services.migrationhub.MigrationHubClient
com.amazonaws.services.mobile.AWSMobileAsyncClient	software.amazon.awssdk.services.mobile.MobileAsyncClient
com.amazonaws.services.mobile.AWSMobileClient	software.amazon.awssdk.services.mobile.MobileClient

1.x 客户端	2.x 客户端
com.amazonaws.services.mq.AmazonMQAsyncClient	software.amazon.awssdk.services.mq.MqAsyncClient
com.amazonaws.services.mq.AmazonMQClient	software.amazon.awssdk.services.mq.MqClient
com.amazonaws.services.mturk.AmazonMTurkAsyncClient	software.amazon.awssdk.services.mturk.MTurkAsyncClient
com.amazonaws.services.mturk.AmazonMTurkClient	software.amazon.awssdk.services.mturk.MTurkClient
com.amazonaws.services.neptune.AmazonNeptuneAsyncClient	software.amazon.awssdk.services.neptune.NeptuneAsyncClient
com.amazonaws.services.neptune.AmazonNeptuneClient	software.amazon.awssdk.services.neptune.NeptuneClient
com.amazonaws.services.opsworks.AWSOpsWorksAsyncClient	software.amazon.awssdk.services.opsworks.OpsWorksAsyncClient
com.amazonaws.services.opsworks.AWSOpsWorksClient	software.amazon.awssdk.services.opsworks.OpsWorksClient
com.amazonaws.services.opsworkscm.AWSOpsWorksCMAsyncClient	software.amazon.awssdk.services.opsworkscm.OpsWorksCmAsyncClient
com.amazonaws.services.opsworkscm.AWSOpsWorksCMClient	software.amazon.awssdk.services.opsworkscm.OpsWorksCmClient
com.amazonaws.services.organizations.AWSOrganizationsAsyncClient	software.amazon.awssdk.services.organizations.OrganizationsAsyncClient

1.x 客户端	2.x 客户端
com.amazonaws.services.organizations.AWSOrganizationsClient	software.amazon.awssdk.services.organizations.OrganizationsClient
com.amazonaws.services.pi.AWSPIAsyncClient	software.amazon.awssdk.services.pi.PiAsyncClient
com.amazonaws.services.pi.AWSPIClient	software.amazon.awssdk.services.pi.PiClient
com.amazonaws.services.pinpoint.AmazonPinpointAsyncClient	software.amazon.awssdk.services.pinpoint.PinpointAsyncClient
com.amazonaws.services.pinpoint.AmazonPinpointClient	software.amazon.awssdk.services.pinpoint.PinpointClient
com.amazonaws.services.polly.AmazonPollyAsyncClient	software.amazon.awssdk.services.polly.PollyAsyncClient
com.amazonaws.services.polly.AmazonPollyClient	software.amazon.awssdk.services.polly.PollyClient
com.amazonaws.services.pricing.AWSPricingAsyncClient	software.amazon.awssdk.services.pricing.PricingAsyncClient
com.amazonaws.services.pricing.AWSPricingClient	software.amazon.awssdk.services.pricing.PricingClient
com.amazonaws.services.rds.AmazonRDSAsyncClient	software.amazon.awssdk.services.rds.RdsAsyncClient
com.amazonaws.services.rds.AmazonRDSClient	software.amazon.awssdk.services.rds.RdsClient

1.x 客户端	2.x 客户端
com.amazonaws.services.redshift.AmazonRedshiftAsyncClient	software.amazon.awssdk.services.redshift.RedshiftAsyncClient
com.amazonaws.services.redshift.AmazonRedshiftClient	software.amazon.awssdk.services.redshift.RedshiftClient
com.amazonaws.services.rekognition.AmazonRekognitionAsyncClient	software.amazon.awssdk.services.rekognition.RekognitionAsyncClient
com.amazonaws.services.rekognition.AmazonRekognitionClient	software.amazon.awssdk.services.rekognition.RekognitionClient
com.amazonaws.services.resourcegroups.AWSResourceGroupsAsyncClient	software.amazon.awssdk.services.resourcegroups.ResourceGroupsAsyncClient
com.amazonaws.services.resourcegroups.AWSResourceGroupsClient	software.amazon.awssdk.services.resourcegroups.ResourceGroupsClient
com.amazonaws.services.resourcegroupstaggingapi.AWSResourceGroupsTaggingAPIAsyncClient	software.amazon.awssdk.services.resourcegroupstaggingapi.ResourceGroupsTaggingApiAsyncClient
com.amazonaws.services.resourcegroupstaggingapi.AWSResourceGroupsTaggingAPIClient	software.amazon.awssdk.services.resourcegroupstaggingapi.ResourceGroupsTaggingApiClient
com.amazonaws.services.route53.AmazonRoute53AsyncClient	software.amazon.awssdk.services.route53.Route53AsyncClient

1.x 客户端	2.x 客户端
com.amazonaws.services.route53.AmazonRoute53Client	software.amazon.awssdk.services.route53.Route53Client
com.amazonaws.services.route53domains.AmazonRoute53DomainsAsyncClient	software.amazon.awssdk.services.route53domains.Route53DomainsAsyncClient
com.amazonaws.services.route53domains.AmazonRoute53DomainsClient	software.amazon.awssdk.services.route53domains.Route53DomainsClient
com.amazonaws.services.s3.AmazonS3Client	software.amazon.awssdk.services.s3.S3Client
com.amazonaws.services.sagemaker.AmazonSageMakerAsyncClient	software.amazon.awssdk.services.sagemaker.SageMakerAsyncClient
com.amazonaws.services.sagemaker.AmazonSageMakerClient	software.amazon.awssdk.services.sagemaker.SageMakerClient
com.amazonaws.services.sagemakerruntime.AmazonSageMakerRuntimeAsyncClient	software.amazon.awssdk.services.sagemakerruntime.SageMakerRuntimeAsyncClient
com.amazonaws.services.sagemakerruntime.AmazonSageMakerRuntimeClient	software.amazon.awssdk.services.sagemakerruntime.SageMakerRuntimeClient
com.amazonaws.services.secretsmanager.AWSSecretsManagerAsyncClient	software.amazon.awssdk.services.secretsmanager.SecretsManagerAsyncClient
com.amazonaws.services.secretsmanager.AWSSecretsManagerClient	software.amazon.awssdk.services.secretsmanager.SecretsManagerClient

1.x 客户端	2.x 客户端
com.amazonaws.services.securitytoken.AWSSecurityTokenServiceAsyncClient	software.amazon.awssdk.services.sts.StsAsyncClient
com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClient	software.amazon.awssdk.services.sts.StsClient
com.amazonaws.services.serverlessapplicationrepository.AWSServerlessApplicationRepositoryAsyncClient	software.amazon.awssdk.services.serverlessapplicationrepository.ServerlessApplicationRepositoryAsyncClient
com.amazonaws.services.serverlessapplicationrepository.AWSServerlessApplicationRepositoryClient	software.amazon.awssdk.services.serverlessapplicationrepository.ServerlessApplicationRepositoryClient
com.amazonaws.services.servermigration.AWSServerMigrationAsyncClient	software.amazon.awssdk.services.sms.SmsAsyncClient
com.amazonaws.services.servermigration.AWSServerMigrationClient	software.amazon.awssdk.services.sms.SmsClient
com.amazonaws.services.servicecatalog.AWSServiceCatalogAsyncClient	software.amazon.awssdk.services.servicecatalog.ServiceCatalogAsyncClient
com.amazonaws.services.servicecatalog.AWSServiceCatalogClient	software.amazon.awssdk.services.servicecatalog.ServiceCatalogClient

1.x 客户端	2.x 客户端
com.amazonaws.services.servicediscovery.AWSServiceDiscoveryAsyncClient	software.amazon.awssdk.services.servicediscovery.ServiceDiscoveryAsyncClient
com.amazonaws.services.servicediscovery.AWSServiceDiscoveryClient	software.amazon.awssdk.services.servicediscovery.ServiceDiscoveryClient
com.amazonaws.services.shield.AWSShieldAsyncClient	software.amazon.awssdk.services.shield.ShieldAsyncClient
com.amazonaws.services.shield.AWSShieldClient	software.amazon.awssdk.services.shield.ShieldClient
com.amazonaws.services.simpledb.AmazonSimpleDBAsyncClient	software.amazon.awssdk.services.simpledb.SimpleDbAsyncClient
com.amazonaws.services.simpledb.AmazonSimpleDBClient	software.amazon.awssdk.services.simpledb.SimpleDbClient
com.amazonaws.services.simpleemail.AmazonSimpleEmailServiceAsyncClient	software.amazon.awssdk.services.ses.SesAsyncClient
com.amazonaws.services.simpleemail.AmazonSimpleEmailServiceClient	software.amazon.awssdk.services.ses.SesClient
com.amazonaws.services.simplesystemsmanagement.AWSSimpleSystemsManagementAsyncClient	software.amazon.awssdk.services.ssm.SsmAsyncClient
com.amazonaws.services.simplesystemsmanagement.AWSSimpleSystemsManagementClient	software.amazon.awssdk.services.ssm.SsmClient

1.x 客户端	2.x 客户端
com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowAsyncClient	software.amazon.awssdk.services.swf.SwfAsyncClient
com.amazonaws.services.simpleworkflow.AmazonSimpleWorkflowClient	software.amazon.awssdk.services.swf.SwfClient
com.amazonaws.services.snowball.AmazonSnowballAsyncClient	software.amazon.awssdk.services.snowball.SnowballAsyncClient
com.amazonaws.services.snowball.AmazonSnowballClient	software.amazon.awssdk.services.snowball.SnowballClient
com.amazonaws.services.sns.AmazonSNSAsyncClient	software.amazon.awssdk.services.sns.SnsAsyncClient
com.amazonaws.services.sns.AmazonSNSClient	software.amazon.awssdk.services.sns.SnsClient
com.amazonaws.services.sqs.AmazonSQSAsyncClient	software.amazon.awssdk.services.sqs.SqsAsyncClient
com.amazonaws.services.sqs.AmazonSQSClient	software.amazon.awssdk.services.sqs.SqsClient
com.amazonaws.services.stepfunctions.AWSStepFunctionsAsyncClient	software.amazon.awssdk.services.sfn.SfnAsyncClient
com.amazonaws.services.stepfunctions.AWSStepFunctionsClient	software.amazon.awssdk.services.sfn.SfnClient

1.x 客户端	2.x 客户端
com.amazonaws.services.storagegateway.AWSStorageGatewayAsyncClient	software.amazon.awssdk.services.storagegateway.StorageGatewayAsyncClient
com.amazonaws.services.storagegateway.AWSStorageGatewayClient	software.amazon.awssdk.services.storagegateway.StorageGatewayClient
com.amazonaws.services.support.AWSSupportAsyncClient	software.amazon.awssdk.services.support.SupportAsyncClient
com.amazonaws.services.support.AWSSupportClient	software.amazon.awssdk.services.support.SupportClient
com.amazonaws.services.transcribe.AmazonTranscribeAsyncClient	software.amazon.awssdk.services.transcribe.TranscribeAsyncClient
com.amazonaws.services.transcribe.AmazonTranscribeClient	software.amazon.awssdk.services.transcribe.TranscribeClient
com.amazonaws.services.translate.AmazonTranslateAsyncClient	software.amazon.awssdk.services.translate.TranslateAsyncClient
com.amazonaws.services.translate.AmazonTranslateClient	software.amazon.awssdk.services.translate.TranslateClient
com.amazonaws.services.waf.AWSWAFAsyncClient	software.amazon.awssdk.services.waf.WafAsyncClient
com.amazonaws.services.waf.AWSWAFClient	software.amazon.awssdk.services.waf.WafClient

1.x 客户端	2.x 客户端
com.amazonaws.services.waf.AWSWAFRegionalAsyncClient	software.amazon.awssdk.services.waf.regional.WafRegionalAsyncClient
com.amazonaws.services.waf.AWSWAFRegionalClient	software.amazon.awssdk.services.waf.regional.WafRegionalClient
com.amazonaws.services.workdocs.AmazonWorkDocsAsyncClient	software.amazon.awssdk.services.workdocs.WorkDocsAsyncClient
com.amazonaws.services.workdocs.AmazonWorkDocsClient	software.amazon.awssdk.services.workdocs.WorkDocsClient
com.amazonaws.services.workmail.AmazonWorkMailAsyncClient	software.amazon.awssdk.services.workmail.WorkMailAsyncClient
com.amazonaws.services.workmail.AmazonWorkMailClient	software.amazon.awssdk.services.workmail.WorkMailClient
com.amazonaws.services.workspaces.AmazonWorkspacesAsyncClient	software.amazon.awssdk.services.workspaces.WorkSpacesAsyncClient
com.amazonaws.services.workspaces.AmazonWorkspacesClient	software.amazon.awssdk.services.workspaces.WorkSpacesClient
com.amazonaws.services.xray.AWSXRayAsyncClient	software.amazon.awssdk.services.xray.XRayAsyncClient
com.amazonaws.services.xray.AWSXRayClient	software.amazon.awssdk.services.xray.XRayClient

客户端创建默认设置

在版本 2.x 中，对默认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括匿名凭证。您必须使用 AnonymousCredentialsProvider 手动指定对 S3 的匿名访问。
- 与默认客户端创建相关的以下环境变量有所不同。

1.x	2.x
AWS_CBOR_DISABLED	CBOR_ENABLED
AWS_ION_BINARY_DISABLE	BINARY_ION_ENABLED

- 与默认客户端创建相关的以下系统属性有所不同。

1.x	2.x
com.amazonaws.sdk.disableEc2Metadata	aws.disableEc2Metadata
com.amazonaws.sdk.ec2MetadataServiceEndpointOverride	aws.ec2MetadataServiceEndpoint
com.amazonaws.sdk.disableCbor	aws.cborEnabled
com.amazonaws.sdk.disableIonBinary	aws.binaryIonEnabled

- 版本 2.x 不支持以下系统属性。
-

1.x
com.amazonaws.sdk.disableCertChecking
com.amazonaws.sdk.enableDefaultMetrics
com.amazonaws.sdk.enableThrottledRetry
com.amazonaws.regions.RegionUtils.fileOverride

1.x
com.amazonaws.regions.RegionUtils.disableRemote
com.amazonaws.services.s3.disableImplicitGlobalClients
com.amazonaws.sdk.enableInRegionOptimizedMode

- 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，已通过在客户端或客户端生成器上设置 ClientConfiguration 实例修改了 SDK 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。利用单独的配置类，您可以为异步客户端与同步客户端配置不同的 HTTP 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 客户端

主要更改

- 在版本 2.x 中，您可以通过使用 `clientBuilder.httpClientBuilder` 指定实现来更改在运行时使用的 HTTP 客户端。
- 使用 `clientBuilder.httpClient` 将 HTTP 客户端传递给服务客户端生成器时，如果服务客户端关闭，则默认情况下不会关闭 HTTP 客户端。这让您可以在服务客户端之间共享 HTTP 客户端。
- 异步 HTTP 客户端现在使用非阻塞 IO。
- 一些操作现在使用 HTTP/2 来提高性能。

更改设置

设置	1.x	2.x 同步，Apache	2.x 异步，Netty
	`ClientConfiguration clientConfig = new ClientConfiguration()`	`ApacheHttpClient.Builder httpClientBuilder =`	`NettyNioAsyncHttpClient.Builder httpClientBuilder =`

设置	1.x	2.x 同步，Apache	2.x 异步，Netty
		`ApacheHttpClient.builder()`	`NettyNioAsyncHttpClient.builder()`
最大连接数	`clientConfig.setMaxConnections(...)` `clientConfig.withMaxConnections(...)`	`httpClientBuilder.maxConnections(...)`	`httpClientBuilder.maxConcurrency(...)`
连接超时	`clientConfig.setConnectionTimeout(...)` `clientConfig.withConnectionTimeout(...)`	`httpClientBuilder.connectionTimeout(...)` `httpClientBuilder.connectionAcquisitionTimeout(...)`	`httpClientBuilder.connectionTimeout(...)`
套接字超时	`clientConfig.setSocketTimeout(...)` `clientConfig.withSocketTimeout(...)`	`httpClientBuilder.socketTimeout(...)`	`httpClientBuilder.writeTimeout(...)` `httpClientBuilder.readTimeout(...)`

设置	1.x	2.x 同步，Apache	2.x 异步，Netty
连接 TTL	`clientConfig.setConnectionTTL(...)` `clientConfig.withConnectionTTL(...)`	`httpClientBuilder.connectionTimeToLive(...)`	`httpClientBuilder.connectionTimeToLive(...)`
连接最大空闲时间	`clientConfig.setConnectionMaxIdleMillis(...)` `clientConfig.withConnectionMaxIdleMillis(...)`	`httpClientBuilder.connectionMaxIdleTime(...)`	`httpClientBuilder.connectionMaxIdleTime(...)`
处于非活动状态后进行验证	`clientConfig.setValidateAfterInactivityMillis(...)` `clientConfig.withValidateAfterInactivityMillis(...)`	不支持（请求功能）	不支持（请求功能）
本地地址	`clientConfig.setLocalAddress(...)` `clientConfig.withLocalAddress(...)`	`httpClientBuilder.localAddress(...)`	不支持

设置	1.x	2.x 同步，Apache	2.x 异步，Netty
已启用 Expect-continue	`clientConfig.setUseExpectContinue(...)` `clientConfig.withUseExpectContinue(...)`	`httpClientBuilder.expectContinueEnabled(...)`	不支持（请求功能）
连接收割者	`clientConfig.setUseReaper(...)` `clientConfig.withReaper(...)`	`httpClientBuilder.useIdleConnectionReaper(...)`	`httpClientBuilder.useIdleConnectionReaper(...)`

	`AmazonDynamoDBClientBuilder.standard()`
	`.withClientConfiguration(clientConfiguration)`
	`.build()`
		`DynamoDbClient.builder()`
		`.httpClientBuilder(httpClientBuilder)`
		`.build()`
			`DynamoDbAsyncClient.builder()`
			`.httpClientBuilder(httpClientBuilder)`
			`.build()`

HTTP 客户端代理

设置	1.x	2.x 同步，Apache	2.x 异步，Netty
	`ClientConfiguration clientConfig =` `new ClientConfiguration()`	`ProxyConfiguration.Builder proxyConfig =`	`ProxyConfiguration.Builder proxyConfig =`

设置	1.x	2.x 同步，Apache	2.x 异步，Netty
		`ProxyConfiguration.builder()`	`ProxyConfiguration.builder()`
代理主机	`clientConfig.setProxyHost(...)` `clientConfig.withProxyHost(...)`	`proxyConfig.endpoint(...)`	`proxyConfig.host(...)`
代理端口	`clientConfig.setProxyPort(...)` `clientConfig.withProxyPort(...)`	`proxyConfig.endpoint(...)` 代理端口嵌入在 endpoint	`proxyConfig.port(...)`
代理用户名	`clientConfig.setProxyUsername(...)` `clientConfig.withProxyUsername(...)`	`proxyConfig.username(...)`	`proxyConfig.username(...)`
代理密码	`clientConfig.setProxyPassword(...)` `clientConfig.withProxyPassword(...)`	`proxyConfig.password(...)`	`proxyConfig.password(...)`

设置	1.x	2.x 同步，Apache	2.x 异步，Netty
代理域	`clientConfig.setProxyDomain(...)` `clientConfig.withProxyDomain(...)`	`proxyConfig.ntlmDomain(...)`	不支持（请求功能）
代理工作站	`clientConfig.setProxyWorkspace(...)` `clientConfig.withProxyWorkstation(...)`	`proxyConfig.ntlmWorkstation(...)`	不支持（请求功能）
代理身份验证方法	`clientConfig.setProxyAuthenticationMethods(...)` `clientConfig.withProxyAuthenticationMethods(...)`	不支持	不支持（请求功能）
预发式基本代理身份验证	`clientConfig.setPreemptiveBasicProxyAuth(...)` `clientConfig.withPreemptiveBasicProxyAuth(...)`	`proxyConfig.preemptiveBasicAuthenticationEnabled(...)`	不支持（请求功能）

设置	1.x	2.x 同步，Apache	2.x 异步，Netty
非代理主机	`clientConfig.setNonProxyHosts(...)` `clientConfig.withNonProxyHosts(...)`	`proxyConfig.nonProxyHosts(...)`	`proxyConfig.nonProxyHosts(...)`
禁用套接字代理	`clientConfig.setDisableSocketProxy(...)` `clientConfig.withDisableSocketProxy(...)`	不支持（请求功能）	不支持（请求功能）

	`AmazonDynamoDBClientBuilder.standard()`
	`.withClientConfiguration(clientConfiguration)`
	`.build()`
		`httpClientBuilder.proxyConfiguration(`
		`proxyConfig.build())`
			`httpClientBuilder.proxyConfiguration(`
			`proxyConfig.build())`

客户端覆盖

设置	1.x	2.x
	`ClientConfiguration clientConfig =`	`ClientOverrideConfiguration.Builder overrideConfig =`

设置	1.x	2.x
	`new ClientConfiguration()`	`ClientOverrideConfiguration.builder()`
用户代理前缀	`clientConfig.setUserAgentPrefix(...)` `clientConfig.withUserAgentPrefix(...)`	`overrideConfig.advancedOption(SdkAdvancedClientOption.USER_AGENT_PREFIX, ...)`
用户代理后缀	`clientConfig.setUserAgentSuffix(...)` `clientConfig.withUserAgentSuffix(...)`	`overrideConfig.advancedOption(SdkAdvancedClientOption.USER_AGENT_SUFFIX, ...)`
Signer	`clientConfig.setSignerOverride(...)` `clientConfig.withSignerOverride(...)`	`overrideConfig.advancedOption(SdkAdvancedClientOption.SIGNER, ...)`
其他标头	`clientConfig.addHeader(...)` `clientConfig.withHeader(...)`	`overrideConfig.putHeader(...)`
请求超时	`clientConfig.setRequestTimeout(...)` `clientConfig.withRequestTimeout(...)`	`overrideConfig.apiCallAttemptTimeout(...)`

设置	1.x	2.x
客户端执行超时	`clientConfig.setClientExecutionTimeout(...)` `clientConfig.withClientExecutionTimeout(...)`	`overrideConfig.apiCallTimeout(...)`
使用 Gzip	`clientConfig.setUseGzip(...)` `clientConfig.withGzip(...)`	不支持（请求功能）
套接字缓冲区大小提示	`clientConfig.setSocketBufferSizeHints(...)` `clientConfig.withSocketBufferSizeHints(...)`	不支持（请求功能）
缓存响应元数据	`clientConfig.setCacheResponseMetadata(...)` `clientConfig.withCacheResponseMetadata(...)`	不支持（请求功能）
响应元数据缓存大小	`clientConfig.setResponseMetadataCacheSize(...)` `clientConfig.withResponseMetadataCacheSize(...)`	不支持（请求功能）

设置	1.x	2.x
DNS 解析程序	`clientConfig.setDnsResolver(...)` `clientConfig.withDnsResolver(...)`	不支持（请求功能）

TCP keepalive	`clientConfig.setUseTcpKeepAlive(...)`
	`clientConfig.withTcpKeepAlive(...)`
		此选项现在位于 HTTP 客户端配置中
		`- ApacheHttpClient.builder().tcpKeepAlive(true)`
		`- NettyNioAsyncHttpClient.builder().tcpKeepAlive(true)`

安全随机	`clientConfig.setSecureRandom(...)` `clientConfig.withSecureRandom(...)`	不支持（请求功能）

	`AmazonDynamoDBClientBuilder.standard()`
	`.withClientConfiguration(clientConfiguration)`
	`.build()`
		`DynamoDbClient.builder()`
		`.overrideConfiguration(overrideConfig.build())`
		`.build()`

客户端覆盖重试次数

设置	1.x	2.x
	`ClientConfiguration clientConfig =` `new ClientConfiguration()`	`ClientOverrideConfiguration.Builder overrideConfigBuilder =`

设置	1.x	2.x
		`ClientOverrideConfiguration.builder();`
最大错误重试数	`clientConfig.setMaxErrorRetry(...)` `clientConfig.withMaxErrorRetry(...)`	`// Configure the default retry strategy.` `overrideConfigBuilder.retryStrategy(b -> b.maxAttempts(...));`
使用带节流控制的重试次数	`clientConfig.setUseThrottleRetries(...)` `clientConfig.withUseThrottleRetries(...)`	不支持
节流前的最大连续重试次数	`clientConfig.setMaxConsecutiveRetriesBeforeThrottling(...)` `clientConfig.withMaxConsecutiveRetriesBeforeThrottling(...)`	不支持

设置	1.x	2.x

	`AmazonDynamoDBClientBuilder.standard()`
	`    .withClientConfiguration(clientConfiguration)`
	`    .build()`
		`DynamoDbClient.builder()`
		`        .overrideConfiguration(overrideConfigBuilder.build())`
		`        .build();`
		
		`// You also have the option to use a lambda expression to configure and`
		`// build the 'ClientOverrideConfiguration.Builder'.`
		`DynamoDbClient client = DynamoDbClient.builder()`
		`        .overrideConfiguration(o -> o.retryStrategy(b -> b.maxAttempts(5)))`
		`        .build();`

异步客户端

设置	1.x	2.x
		`ClientAsyncConfiguration.Builder asyncConfig = ClientAsyncConfiguration.builder()`
执行程序	`AmazonDynamoDBAsyncClientBuilder.standard()`	`asyncConfig.advancedOption(`

设置	1.x	2.x
	`.withExecutorFactory(...)` `.build()`	`SdkAdvancedAsyncClientOption.FUTURE_COMPLETION_EXECUTOR, ...)`

		`DynamoDbAsyncClient.builder()`
		`.asyncConfiguration(asyncConfig)`
		`.build()`

其他客户端更改

1.x 中的以下 `ClientConfiguration` 选项在 SDK 的 2.x 中已更改，没有直接等效项。

设置	1.x	2.x 等效项

协议	`clientConfig.setProtocol(Protocol.HTTP)`
	`clientConfig.withProtocol(Protocol.HTTP)`
		默认情况下，协议设置为 HTTPS。要修改设置，请在客户端生成器中指定协议设置为 HTTP 端点：
		`clientBuilder.endpointOverride(`
		`URI.create("http://..."))`

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/类名称	`com.amazonaws.auth.AWSCredentialsProvider`	`software.amazon.awssdk.auth.credentials.AwsCredentialsProvider`
方法名称	`getCredentials`	`resolveCredentials`
不受支持的方法	`refresh`	不支持

DefaultAWSCredentialsProviderChain

更改类别	1.x	2.x
软件包/类名称	com.amazonaws.auth.DefaultAWSCredentialsProviderChain	software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider
创建	new DefaultAWSCredentialsProviderChain	DefaultCredentialsProvider.create
不受支持的方法	getInstance	不支持
外部设置的优先顺序	环境变量在系统属性之前	系统属性在环境变量之前

AWSStaticCredentialsProvider

更改类别	1.x	2.x
软件包/类名称	com.amazonaws.auth.AWSStaticCredentialsProvider	software.amazon.awssdk.auth.credentials.StaticCredentialsProvider
创建	new AWSStaticCredentialsProvider	StaticCredentialsProvider.create

EnvironmentVariableCredentialsProvider

更改类别	1.x	2.x
软件包/类名称	com.amazonaws.auth.EnvironmentVariableCredentialsProvider	software.amazon.awssdk.auth.credentials.EnvironmentVar

更改类别	1.x	2.x
		iableCredentialsProvider
创建	new EnvironmentVariableCredentialsProvider	EnvironmentVariableCredentialsProvider.create
环境变量名称	AWS_ACCESS_KEY	AWS_ACCESS_KEY_ID
	AWS_SECRET_KEY	AWS_SECRET_ACCESS_KEY

SystemPropertiesCredentialsProvider

更改类别	1.x	2.x
软件包/类名称	com.amazonaws.auth.SystemPropertiesCredentialsProvider	software.amazon.awssdk.auth.credentials.SystemPropertyCredentialsProvider
创建	new SystemPropertiesCredentialsProvider	SystemPropertiesCredentialsProvider.create
系统属性名称	aws.secretKey	aws.secretAccessKey

ProfileCredentialsProvider

更改类别	1.x	2.x
软件包/类名称	com.amazonaws.auth.profile.ProfileCredentialsProvider	software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider

更改类别	1.x	2.x
创建	`new ProfileCredentialsProvider`	`ProfileCredentialsProvider.create`
自定义配置文件的位置	• `AWS_CREDENTIAL_PROFILES_FILE` 环境变量 • `new ProfileCredentialsProvider`	• `AWS_SHARED_CREDENTIALS_FILE` 环境变量 • `ProfileCredentialsProvider.builder`

ContainerCredentialsProvider

更改类别	1.x	2.x
软件包/类名称	`com.amazonaws.auth.ContainerCredentialsProvider`	`software.amazon.awssdk.auth.credentials.ContainerCredentialsProvider`
创建	`new ContainerCredentialsProvider`	`ContainerCredentialsProvider.create`
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/类名称	`com.amazonaws.auth.InstanceProfileCredentialsProvider`	`software.amazon.awssdk.auth.credentials.InstanceProfileCredentialsProvider`

更改类别	1.x	2.x
创建	`new InstanceProfileCredentialsProvider`	`InstanceProfileCredentialsProvider.create`
指定异步刷新	`new InstanceProfileCredentialsProvider(true)`	`InstanceProfileCredentialProvider.builder().asyncCredentialUpdateEnabled(true).build()`
系统属性名称	`com.amazonaws.sdk.disableEc2Metadata`	`aws.disableEc2Metadata`
	`com.amazonaws.sdk.ec2MetadataServiceEndpointOverride`	`aws.ec2MetadataServiceEndpoint`

STSAssumeRoleSessionCredentialsProvider

更改类别	1.x	2.x
软件包/类名称	`com.amazonaws.auth.STSAssumeRoleSessionCredentialsProvider`	`software.amazon.awssdk.services.sts.auth.StsAssumeRoleCredentialsProvider`
创建	• `new STSAssumeRoleSessionCredentialsProvider` • `new STSAssumeRoleSessionCredentialsProvider.Builder`	`StsAssumeRoleCredentialsProvider.builder`
异步刷新	默认行为	默认行为

更改类别	1.x	2.x
配置	new STSAssumeRoleSessionCredentialsProvider.Builder	配置 StsClient 和 AssumeRoleRequest 请求

STSSessionCredentialsProvider

更改类别	1.x	2.x
软件包/类名称	com.amazonaws.auth.STSSessionCredentialsProvider	software.amazon.awssdk.services.sts.auth.StsGetSessionTokenCredentialsProvider
创建	new STSSessionCredentialsProvider	StsGetSessionTokenCredentialsProvider.builder
异步刷新	默认行为	StsGetSessionTokenCredentialsProvider.builder
配置	构造函数参数	在生成器中配置 StsClient 和 GetSessionTokenRequest 请求

WebIdentityFederationSessionCredentialsProvider

更改类别	1.x	2.x
软件包/类名称	com.amazonaws.auth.WebIdentityFedera	software.amazon.awssdk.services.sts.auth.StsAssumeRole

更改类别	1.x	2.x
	tionSessionCredentialsProvider	WithWebIdentityCredentialsProvider
创建	new WebIdentityFederationSessionCredentialsProvider	StsAssumeRoleWithWebIdentityCredentialsProvider.builder
异步刷新	默认行为	StsAssumeRoleWithWebIdentityCredentialsProvider.builder
配置	构造函数参数	在生成器中配置 StsClient 和 AssumeRoleWithWebIdentityRequest 请求

已替换的类

1.x 类	2.x 替换类
com.amazonaws.auth.EC2ContainerCredentialsProviderWrapper	software.amazon.awssdk.auth.credentials.ContainerCredentialsProvider 和 software.amazon.awssdk.auth.credentials.InstanceProfileCredentialsProvider
com.amazonaws.services.s3.S3CredentialsProviderChain	software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider 和 software.amazon.awssdk.auth.credentials.AnonymousCredentialsProvider

已移除的类

1.x 类
com.amazonaws.auth.ClasspathPropertiesFileCredentialsProvider
com.amazonaws.auth.PropertiesFileCredentialsProvider

区域更改

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
Regions.fromName	Region.of

1.x	2.x
`Regions.getName`	`Region.id`
`Regions.getDescription`	`Region.metadata().description()`
`Regions.getCurrentRegion`	不支持
`Regions.DEFAULT_REGION`	不支持
`Regions.name`	`Region.id`

1.x Region 类方法更改

1.x	2.x
`Region.getName`	`Region.id`
`Region.hasHttpsEndpoint`	不支持
`Region.hasHttpEndpoint`	不支持
`Region.getAvailableEndpoints`	不支持
`Region.createClient`	不支持

RegionMetadata 类方法更改

1.x	2.x
`RegionMetadata.getName`	`RegionMetadata.name`
`RegionMetadata.getDomain`	`RegionMetadata.domain`
`RegionMetadata.getPartition`	`RegionMetadata.partition`

ServiceMetadata 类方法更改

1.x	2.x
`Region.getServiceEndpoint`	`ServiceMetadata.endpointFor(Region)`
`Region.isServiceSupported`	`ServiceMetadata.regions().contains(Region)`

操作、请求和响应更改

在适用于 Java 的 SDK 版本 2 中，请求会传递给客户端操作。例如 `DynamoDbClient's PutItemRequest` 传递给 `DynamoDbClient.putItem` 操作。这些操作会返回来自 AWS 服务的响应，例如 `PutItemResponse`。

适用于 Java 的 SDK 版本 2 与版本 1 相比有以下变化。

- 现在，具有多个响应页面的操作使用 `Paginator` 方法来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，版本 1 的 `new PutItemRequest().withTableName(...)` 现在为 `PutItemRequest.builder().tableName(...).build()`。
- 操作支持使用简写方式来创建请求：`dynamoDbClient.putItem(request -> request.tableName(...))`。

以下各节说明版本 1 和版本 2 之间的具体更改。有些参数类型更改可以使用迁移工具自动转换，而其他更改则需要手动更新代码。

Date 参数更改

在版本 1 中，许多操作在处理基于时间的参数时都接受 `java.util.Date` 对象。在版本 2 中，这些操作改为使用 `java.time.Instant` 对象。

您可以使用迁移工具自动转换 `Date` 参数，也可以通过在 `Date` 对象上调用 `toInstant()` 方法来手动转换参数。

Example - 在版本 1 中，使用 Date 指定过期日期来生成预签名 URL

```
// Generate a presigned URL that expires at a specific date
Date expiration = new Date(System.currentTimeMillis() + 3600000); // 1 hour from now
URL presignedUrl = s3Client.generatePresignedUrl(bucketName, keyName, expiration);
```

Example - 在版本 2 中，使用 Instant 指定过期时间来生成预签名 URL

```
// Generate a presigned URL that expires at a specific instant
Date expiration = new Date(System.currentTimeMillis() + 3600000); // 1 hour from now
PresignedGetObjectRequest presignedRequest = presigner.presignGetObject(
    GetObjectPresignRequest.builder()
        .getObjectRequest(GetObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build())
        .signatureDuration(Duration.between(Instant.now(), expiration.toInstant()))
        .build());
```

二进制数据处理更改

在版本 1 中，使用 ByteBuffer 对象直接处理二进制数据。在版本 2 中，SDK 使用 SdkBytes 对象，该对象提供一种更便捷、类型更安全的方式来处理二进制数据。

您可以使用迁移工具自动将 SdkBytes 转换为 ByteBuffer，也可以通过在返回的 SdkBytes 对象上调用 asByteBuffer() 来手动转换参数。

Example - 在版本 1 中从消息属性获取二进制数据

```
// Get binary data from a message attribute
MessageAttributeValue messageAttributeValue = new MessageAttributeValue();
ByteBuffer binaryValue = messageAttributeValue.getBinaryValue();
String binaryString = new String(messageAttributeValue.getBinaryValue().array());
```

Example - 在版本 2 中从消息属性获取二进制数据

```
// Get binary data from a message attribute
MessageAttributeValue messageAttributeValue = MessageAttributeValue.builder().build();
ByteBuffer binaryValue = messageAttributeValue.binaryValue().asByteBuffer();
String binaryString = new
 String(messageAttributeValue.binaryValue().asByteBuffer().array());
```

超时参数更改

在版本 1 中，超时值指定为代表毫秒的整数值。在版本 2 中，超时参数使用 `java.time.Duration` 对象来获得更好的类型安全性和清晰度。

您可以使用迁移工具自动转换数值超时值，也可以通过使用适当的 `Duration` 工厂方法来包装数值，从而手动转换这些值。

Example - 在版本 1 中设置请求超时

```
// Set request timeout in milliseconds
ClientConfiguration clientConfiguration = new ClientConfiguration();
clientConfiguration.setRequestTimeout(5000); // 5 seconds
```

Example - 在版本 2 中设置请求超时

```
// Set request timeout using Duration
ClientConfiguration clientConfiguration = new ClientConfiguration();
clientConfiguration.setRequestTimeout(Duration.ofMillis(5000)); // 5 seconds

// Or more clearly:
clientConfiguration.setRequestTimeout(Duration.ofSeconds(5)); // 5 seconds
```

您可以使用以下 `Duration` 工厂方法设置超时值：

- `Duration.ofMillis(long millis)` - 用于毫秒值。
- `Duration.ofSeconds(long seconds)` - 用于秒值。
- `Duration.ofMinutes(long minutes)` - 用于分钟值。

AWS SDK for Java 1.x 和 2.x 之间的流式操作差异

流式操作（例如 Amazon S3 `getObject` 和 `putObject` 方法）支持 SDK 版本 2.x 中的非阻塞 I/O。因此，请求和响应模型对象不再获取 `InputStream` 作为参数。相反，对于异步请求，请求对象接受 `RequestBody`（这是一个字节流）。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")));
```

在 V2 中，流式响应对象接受 ResponseTransformer（对于同步客户端）和 AsyncResponseTransformer（对于异步客户端）。

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")));
```

在适用于 Java 的 SDK 2.x 中，流式响应操作使用 AsBytes 方法将响应加载到内存中，并简化了内存中的常见类型转换。

AWS SDK for Java 1.x 和 2.x 之间的序列化差异

列出请求对象的参数差异

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=
```

对比 V1 中的 POJO 与 V2 中的生成器

由于适用于 Java 的 SDK V1 使用可变的 POJO 类，因此序列化和反序列化库（例如 Jackson）可以直接使用模型对象。

相比之下，适用于 Java 的 SDK V2 使用不可变的模型对象。必须使用中间生成器来执行反序列化/序列化。

以下示例显示了 V1 和 V2 使用 Jackson `ObjectMapper` 对 `headBucket` API 调用进行反序列化/序列化的差异。

```
    public void sendRequest() throws IOException {
        final String bucketName = "amzn-s3-demo-bucket";
        final ObjectMapper mapper = new ObjectMapper();

        // V1 uses POJOs to serialize and deserialize.
        final AmazonS3 v1S3Client = AmazonS3ClientBuilder.defaultClient();
        HeadBucketResult resultV1 = v1S3Client.headBucket(
            new HeadBucketRequest(bucketName));

        String v1Serialized = mapper.writeValueAsString(resultV1);

        HeadBucketResult deserializedV1 = mapper.readValue(v1Serialized,
 HeadBucketResult.class);

        // V2 uses builders to serialize and deserialize.
        S3Client v2S3Client = S3Client.create();
        HeadBucketResponse v2Response = v2S3Client.headBucket(
```

```
            b -> b.bucket(bucketName));

        String v2Serialized = mapper.writeValueAsString(
            v2Response.toBuilder());

        HeadBucketResponse v2Deserialized = mapper.readValue(
            v2Serialized, HeadBucketResponse.serializableBuilderClass())
            .build();
    }
```

AWS SDK for Java 1.x 和 2.x 之间的反序列化差异

V2 中的空集合与 V1 中的 **nulls** 相比

适用于 Java 的 SDK v1.x 和 v2.x 在反序列化包含空列表或映射的 JSON 响应时行为不同。

当 SDK 收到缺失属性的响应时，该响应在模型中定义为列表或映射，V1 会将缺失的属性反序列化为 `null`，而 V2 则将属性反序列化为不可变的空集合对象。

例如，考虑从 DynamoDB 客户端为 `describeTable` 方法返回的响应。以下示例方法包含一个 V2 DynamoDB 客户端和一个 V1 DynamoDB 客户端，它们都在没有全局二级索引的表上执行 `describeTable` 方法

Example在响应中缺失、但在模型中定义为列表类型的属性的反序列化

```
public void deserializationDiffs(){

    DescribeTableResponse v2Response = dynamoDbClientV2.describeTable(builder ->
 builder.tableName(TABLE_NAME));
    // V2 provides has* methods on model objects for list/map members. No null check
 needed.
    LOGGER.info( String.valueOf(v2Response.table().hasGlobalSecondaryIndexes()) );

 LOGGER.info( String.valueOf(v2Response.table().globalSecondaryIndexes().isEmpty()) );
    // V2 deserialize to an empty collection.
    LOGGER.info(v2Response.table().globalSecondaryIndexes().toString());

    // V1 deserialize to null.
    DescribeTableResult v1Result = dynamoDbClientV1.describeTable(new
 DescribeTableRequest(TABLE_NAME));
    if (v1Result.getTable().getGlobalSecondaryIndexes() != null){
        LOGGER.info(v1Result.getTable().getGlobalSecondaryIndexes().toString());
    } else {
```

```
        LOGGER.info("The list of global secondary indexes returned by the V1 call is
 <null>");
    }
}
```

以下内容展示了记录的日志输出：

```
INFO  org.example.DeserializationDifferences:45 - false
INFO  org.example.DeserializationDifferences:46 - true
INFO  org.example.DeserializationDifferences:48 - []
INFO  org.example.DeserializationDifferences:55 - The list of global secondary indexes
 returned by the V1 call is <null>
```

Java SDK 2.x 采用了一种惯用的方法，将空列表和映射反序列化为不可变的空集合，而不是返回 null，从而有助于编写更安全、更简洁的代码。在 V2 中，您可以使用 has* 方法（例如，前面的示例中所示的 hasGlobalSecondaryIndexes）检查服务是否返回了在模型中定义为列表或映射的属性。

使用这种方法，就不需要进行明确的 null 检查，并且符合现代 Java 中处理“缺失”或“空”数据结构的最佳实践。

完整示例

```
package org.example;

import com.amazonaws.regions.Regions;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;
import com.amazonaws.services.dynamodbv2.model.DescribeTableRequest;
import com.amazonaws.services.dynamodbv2.model.DescribeTableResult;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.dynamodb.DynamoDbClient;
import software.amazon.awssdk.services.dynamodb.model.BillingMode;
import software.amazon.awssdk.services.dynamodb.model.DescribeTableResponse;
import software.amazon.awssdk.services.dynamodb.model.KeyType;
import software.amazon.awssdk.services.dynamodb.model.ScalarAttributeType;

import java.util.UUID;

public class DeserializationDifferences {
```

```
   private static final Logger LOGGER =
LoggerFactory.getLogger(DeserializationDifferences.class);
   private static final String TABLE_NAME = "DeserializationTable-" +
UUID.randomUUID();
   DynamoDbClient dynamoDbClientV2 = DynamoDbClient.create();
   AmazonDynamoDB dynamoDbClientV1 =
AmazonDynamoDBClientBuilder.standard().withRegion(Regions.US_EAST_1).build();

   public static void main(String[] args) {

       DeserializationDifferences difference = new DeserializationDifferences();
       difference.createTable();
       difference.deserializationDiffs();
       difference.deleteTable();
   }

   public void createTable(){
       dynamoDbClientV2.createTable(b -> b
               .billingMode(BillingMode.PAY_PER_REQUEST)
               .tableName(TABLE_NAME)
               .keySchema(b1 -> b1.attributeName("Id").keyType(KeyType.HASH))
               .attributeDefinitions(b2 ->
b2.attributeName("Id").attributeType(ScalarAttributeType.S)));
       dynamoDbClientV2.waiter().waitUntilTableExists(b -> b.tableName(TABLE_NAME));
   }

   public void deserializationDiffs(){

       DescribeTableResponse v2Response = dynamoDbClientV2.describeTable(builder ->
builder.tableName(TABLE_NAME));
       // V2 provides has* methods on model objects for list/map members. No null
check needed.
       LOGGER.info( String.valueOf(v2Response.table().hasGlobalSecondaryIndexes()) );

LOGGER.info( String.valueOf(v2Response.table().globalSecondaryIndexes().isEmpty()) );
       // V2 deserialize to an empty collection.
       LOGGER.info(v2Response.table().globalSecondaryIndexes().toString());

       // V1 deserialize to null.
       DescribeTableResult v1Result = dynamoDbClientV1.describeTable(new
DescribeTableRequest(TABLE_NAME));
       if (v1Result.getTable().getGlobalSecondaryIndexes() != null){
           LOGGER.info(v1Result.getTable().getGlobalSecondaryIndexes().toString());
       } else {
```

```
            LOGGER.info("The list of global secondary indexes returned by the V1 call
 is <null>");
        }
    }

    public void deleteTable(){
        dynamoDbClientV2.deleteTable(b -> b.tableName(TABLE_NAME));
        dynamoDbClientV2.waiter().waitUntilTableNotExists(b ->
 b.tableName(TABLE_NAME));
    }
}
```

来自 V1 和 V2 客户端的 `describeTable` 方法的 JSON 响应不包含 `GlobalSecondaryIndexes` 属性：

```
{
  "Table": {
    "AttributeDefinitions": [
      {
        "AttributeName": "Id",
        "AttributeType": "S"
      }
    ],
    "BillingModeSummary": {
      "BillingMode": "PAY_PER_REQUEST",
      "LastUpdateToPayPerRequestDateTime": ...
    },
    "CreationDateTime": ...,
    "DeletionProtectionEnabled": false,
    "ItemCount": 0,
    "KeySchema": [
      {
        "AttributeName": "Id",
        "KeyType": "HASH"
      }
    ],
    "ProvisionedThroughput": {
      "NumberOfDecreasesToday": 0,
      "ReadCapacityUnits": 0,
      "WriteCapacityUnits": 0
    },
    "TableArn": "arn:aws:dynamodb:us-east-1:11111111111:table/
DeserializationTable-...",
```

```
    "TableId": "...",
    "TableName": "DeserializationTable-...",
    "TableSizeBytes": 0,
    "TableStatus": "ACTIVE",
    "TableThroughputModeSummary": {
      "LastUpdateToPayPerRequestDateTime": ...,
      "TableThroughputMode": "PAY_PER_REQUEST"
    },
    "WarmThroughput": {
      "ReadUnitsPerSecond": 12000,
      "Status": "ACTIVE",
      "WriteUnitsPerSecond": 4000
    }
  }
}
```

Exception 更改

异常类名及其结构和关系已更改。`software.amazon.awssdk.core.exception.SdkException` 是新的基础 `Exception` 类，所有其他异常从该类进行扩展。

此表列出了 Exception 类名更改的映射。

1.x	2.x
`com.amazonaws.SdkBaseException com.amazonaws.AmazonClientException`	`software.amazon.awssdk.core.exception.SdkException`
`com.amazonaws.SdkClientException`	`software.amazon.awssdk.core.exception.SdkClientException`
`com.amazonaws.AmazonServiceException`	`software.amazon.awssdk.awscore.exception.AwsServiceException`

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
`AmazonServiceException.getRequestId`	`SdkServiceException.requestId`
`AmazonServiceException.getServiceName`	`AwsServiceException.awsErrorDetails().serviceName`
`AmazonServiceException.getErrorCode`	`AwsServiceException.awsErrorDetails().errorCode`
`AmazonServiceException.getErrorMessage`	`AwsServiceException.awsErrorDetails().errorMessage`
`AmazonServiceException.getStatusCode`	`AwsServiceException.awsErrorDetails().sdkHttpResponse().statusCode`
`AmazonServiceException.getHttpHeaders`	`AwsServiceException.awsErrorDetails().sdkHttpResponse().headers`
`AmazonServiceException.rawResponse`	`AwsServiceException.awsErrorDetails().rawResponse`

特定于服务的更改

Amazon S3 更改

适用于 Java 的 SDK 2.x 默认禁用匿名访问。因此，您必须使用 `AnonymousCredentialsProvider` 启用匿名访问。

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。

下表显示两个版本中的操作名称。

Amazon S3 操作名称

1.x	2.x
abortMultipartUpload	abortMultipartUpload
changeObjectStorageClass	copyObject
completeMultipartUpload	completeMultipartUpload
copyObject	copyObject
copyPart	uploadPartCopy
createBucket	createBucket
deleteBucket	deleteBucket
deleteBucketAnalyticsConfiguration	deleteBucketAnalyticsConfiguration
deleteBucketCrossOriginConfiguration	deleteBucketCors
deleteBucketEncryption	deleteBucketEncryption
deleteBucketInventoryConfiguration	deleteBucketInventoryConfiguration
deleteBucketLifecycleConfiguration	deleteBucketLifecycle
deleteBucketMetricsConfiguration	deleteBucketMetricsConfiguration
deleteBucketPolicy	deleteBucketPolicy
deleteBucketReplicationConfiguration	deleteBucketReplication
deleteBucketTaggingConfiguration	deleteBucketTagging
deleteBucketWebsiteConfiguration	deleteBucketWebsite

1.x	2.x
deleteObject	deleteObject
deleteObjectTagging	deleteObjectTagging
deleteObjects	deleteObjects
deleteVersion	deleteObject
disableRequesterPays	putBucketRequestPayment
doesBucketExist	headBucket
doesBucketExistV2	headBucket
doesObjectExist	headObject
enableRequesterPays	putBucketRequestPayment
generatePresignedUrl	S3Presigner
getBucketAccelerateConfiguration	getBucketAccelerateConfiguration
getBucketAcl	getBucketAcl
getBucketAnalyticsConfiguration	getBucketAnalyticsConfiguration
getBucketCrossOriginConfiguration	getBucketCors
getBucketEncryption	getBucketEncryption
getBucketInventoryConfiguration	getBucketInventoryConfiguration
getBucketLifecycleConfiguration	getBucketLifecycle 或者getBucketLifecycleConfiguration
getBucketLocation	getBucketLocation
getBucketLoggingConfiguration	getBucketLogging

1.x	2.x
getBucketMetricsConfiguration	getBucketMetricsConfiguration
getBucketNotificationConfiguration	getBucketNotification 或者getBucketNotificationConfiguration
getBucketPolicy	getBucketPolicy
getBucketReplicationConfiguration	getBucketReplication
getBucketTaggingConfiguration	getBucketTagging
getBucketVersioningConfiguration	getBucketVersioning
getBucketWebsiteConfiguration	getBucketWebsite
getObject	getObject
getObjectAcl	getObjectAcl
getObjectAsString	getObjectAsBytes().asUtf8String
getObjectMetadata	headObject
getObjectTagging	getObjectTagging
getResourceUrl	S3Utilities#getUrl
getS3AccountOwner	listBuckets
getUrl	S3Utilities#getUrl
headBucket	headBucket
initiateMultipartUpload	createMultipartUpload
isRequesterPaysEnabled	getBucketRequestPayment

1.x	2.x
listBucketAnalyticsConfigurations	listBucketAnalyticsConfigurations
listBucketInventoryConfigurations	listBucketInventoryConfigurations
listBucketMetricsConfigurations	listBucketMetricsConfigurations
listBuckets	listBuckets
listMultipartUploads	listMultipartUploads
listNextBatchOfObjects	listObjectsV2Paginator
listNextBatchOfVersions	listObjectVersionsPaginator
listObjects	listObjects
listObjectsV2	listObjectsV2
listParts	listParts
listVersions	listObjectVersions
putObject	putObject
restoreObject	restoreObject
restoreObjectV2	restoreObject
selectObjectContent	selectObjectContent
setBucketAccelerateConfiguration	putBucketAccelerateConfiguration
setBucketAcl	putBucketAcl
setBucketAnalyticsConfiguration	putBucketAnalyticsConfiguration

1.x	2.x
setBucketCrossOriginConfiguration	putBucketCors
setBucketEncryption	putBucketEncryption
setBucketInventoryConfiguration	putBucketInventoryConfiguration
setBucketLifecycleConfiguration	putBucketLifecycle 或者putBucketLifecycleConfiguration
setBucketLoggingConfiguration	putBucketLogging
setBucketMetricsConfiguration	putBucketMetricsConfiguration
setBucketNotificationConfiguration	putBucketNotification 或者putBucketNotificationConfiguration
setBucketPolicy	putBucketPolicy
setBucketReplicationConfiguration	putBucketReplication
setBucketTaggingConfiguration	putBucketTagging
setBucketVersioningConfiguration	putBucketVersioning
setBucketWebsiteConfiguration	putBucketWebsite
setObjectAcl	putObjectAcl
setObjectRedirectLocation	copyObject
setObjectTagging	putObjectTagging
uploadPart	uploadPart

Amazon SNS 更改

除配置为可访问的区域外，SNS 客户端无法再访问其他区域中的 SNS 主题。

Amazon SQS 更改

除配置为可访问的区域外，SQS 客户端无法再访问其他区域中的 SQS 队列。

Amazon RDS 更改

适用于 Java 的 SDK 2.x 使用 `RdsUtilities#generateAuthenticationToken` 代替 1.x 中的类 `RdsIamAuthTokenGenerator`。

从AWS SDK for Java 版本 1 升级到版本 2 时使用 Amazon S3 所发生的更改

AWS SDK for Java 2.x 对 S3 客户端进行了重大更改，包括新的包结构、更新的类名和修订的方法签名。虽然使用迁移工具可以自动将许多方法从 V1 迁移到 V2，但有些方法需要手动迁移，例如 `listNextBatchOfObjects` 和 `selectObjectContent`。此外，V2 用新的实现取代了某些 V1 类（例如 `AccessControlList` 和 `CannedAccessControlList`）。

主题

- AWS SDK for Java 版本 1 和版本 2 之间的 S3 客户端差异
- 将 Transfer Manager 从AWS SDK for Java 的版本 1 迁移到版本 2。
- 从版本 1 升级到版本 2 时解析 Amazon S3 URI 发生的更改
- S3 事件通知 API 从版本 1 到版本 2 的更改

AWS SDK for Java 版本 1 和版本 2 之间的 S3 客户端差异

在本主题中，适用于 Java 的 SDK 版本 1 和版本 2 中的 S3 客户端之间的差异按迁移工具的自动迁移方式进行组织。该工具支持将大多数方法从 V1 迁移到 V2，但有些方法需要手动迁移。除了 S3 客户端方法外，某些 S3 V1 类没有直接的 V2 等效项，需要您手动迁移这些方法。

目录

- 迁移工具支持的 V1 方法示例
  - putObject
  - getObject

- 需要手动迁移的 V1 方法
  - V1 的 getObject 使用 V1 的 S3ObjectId 到 V2 的 GetObjectRequest.builder()
  - V1 的 listNextBatchOfObjects 到 V2 的 listObjectsV2Paginator
  - V1 的 listNextBatchOfVersions 到 V2 的 listObjectVersionsPaginator
  - selectObjectContent
  - V1 的 setBucketAcl 到 V2 在 PutBucketAclRequest.builder() 上的 acl 方法
  - V1 的 setObjectAcl 到 V2 在 PutObjectAclRequest.builder() 上的 acl 方法
  - V1 的 initiateMultipartUpload 到 V2 的 createMultipartUpload
    - 示例迁移
    - 实施差异
  - V1 的 setRegion 到 V2 在客户端生成器上的 region 方法
  - V1 的 setS3ClientOptions(S3ClientOptions clientOptions)
  - V1 的 setBucketLoggingConfiguration 到 V2 的 putBucketLogging
  - V1 的 setBucketLifecycleConfiguration 到 V2 的 putBucketLifecycleConfiguration
  - V1 的 setBucketTaggingConfiguration 到 V2 的 putBucketTagging
- 需要手动迁移的 V1 类
  - V1 的 AccessControlList 到 V2 的 AccessControlPolicy
  - V1 的 CannedAccessControlList 枚举到 V2 的 BucketCannedACL 和 ObjectCannedACL 枚举
  - V1 的 BucketNotificationConfiguration 到 V2 的 NotificationConfiguration
  - V1 的 MultiFactorAuthentication 到 V2 在请求生成器上的 mfa 方法

迁移工具支持的 V1 方法示例

迁移工具可自动将大多数方法从 V1 迁移到 V2。`putObject` 和 `getObject` 方法就是其中的两个示例。

`putObject`

```
// SDK V1
s3Client.putObject("amzn-s3-demo-bucket", "my-key", "Hello World!");


// SDK V2
s3Client.putObject(req -> req
```

```
    .bucket("amzn-s3-demo-bucket")
    .key("my-key"),
    RequestBody.fromString("Hello World!"));
```

getObject

```
// SDK V1
S3Object object = s3Client.getObject("amzn-s3-demo-bucket", "my-key");
InputStream content = object.getObjectContent();


// SDK V2
ResponseInputStream<GetObjectResponse> response = s3Client.getObject(req -> req
    .bucket("amzn-s3-demo-bucket")
    .key("my-key"));
```

需要手动迁移的 V1 方法

您需要手动迁移以下 V1 S3 客户端方法。使用迁移工具时，您会在 V2 输出 Java 文件中看到一条注释，该注释会引导您进入此主题。

V1 的 **getObject** 使用 V1 的 **S3ObjectId** 到 V2 的 **GetObjectRequest.builder()**

```
// SDK V1
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard()
        .withRegion(Regions.US_WEST_2)
        .build();

S3ObjectId s3ObjectId = new S3ObjectId(
    "amzn-s3-demo-bucket",
    "object-key",
    "abc123version"
);

GetObjectRequest getRequest= new GetObjectRequest(s3ObjectId);
S3Object s3ObjectVersioned = s3Client.getObject(getRequest);


// SDK V2
// V2 does not include a 'S3ObjectId' class. V2 uses the request builder pattern
// to supply the bucket, key, and version parameters.
S3Client s3Client = S3Client.builder()
    .region(Region.US_WEST_2)
```

```
    .build();

GetObjectRequest getRequest = GetObjectRequest.builder()
    .bucket("amzn-s3-demo-bucket")
    .key("object-key")
    .versionId("abc123version")
    .build();

ResponseInputStream<GetObjectResponse> response = s3Client.getObject(getRequest);
```

V1 的 **listNextBatchOfObjects** 到 V2 的 **listObjectsV2Paginator**

```
// SDK V1
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .build();

ObjectListing objectListing = s3ClientV1.listObjects("bucket-name");
while (objectListing.isTruncated()) {
    objectListing = s3ClientV1.listNextBatchOfObjects(objectListing);
    for (S3ObjectSummary summary : objectListing.getObjectSummaries()) {
        System.out.println(summary.getKey());
    }
}


// SDK V2
S3Client s3ClientV2 = S3Client.builder()
    .region(Region.US_WEST_2)
    .build();

ListObjectsV2Request request = ListObjectsV2Request.builder()
    .bucket("bucket-name")
    .build();

// V2 returns a paginator.
ListObjectsV2Iterable responses = s3Client.listObjectsV2Paginator(request);

for (ListObjectsV2Response page : responses) {
    page.contents().forEach(content -> {
        System.out.println(content.key());
    });
}
```

V1 的 **listNextBatchOfVersions** 到 V2 的 **listObjectVersionsPaginator**

```
// SDK V1
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .build();

VersionListing versionListing = s3ClientV1.listVersions("bucket-name", "prefix");
while (versionListing.isTruncated()) {
    versionListing = s3ClientV1.listNextBatchOfVersions(versionListing);
    for (S3VersionSummary version : versionListing.getVersionSummaries()) {
        System.out.println(version.getKey() + " " + version.getVersionId());
    }
}


// SDK V2
S3Client s3ClientV2 = S3Client.builder()
    .region(Region.US_WEST_2)
    .build();

ListObjectVersionsRequest request = ListObjectVersionsRequest.builder()
    .bucket("bucket-name")
    .prefix("prefix")
    .build();

// V2 returns a paginator.
ListObjectVersionsIterable responses = s3ClientV2.listObjectVersionsPaginator(request);

for (ListObjectVersionsResponse page : responses) {
    page.versions().forEach(version -> {
        System.out.println(version.key() + " " + version.versionId());
    });
}
```

selectObjectContent

```
// SDK V1
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .build();

SelectObjectContentRequest request = new SelectObjectContentRequest()
```

```
    .withBucket("bucket-name")
    .withKey("object-key")
    .withExpression("select * from S3Object")
    .withExpressionType(ExpressionType.SQL)


SelectObjectContentResult result = s3ClientV1.selectObjectContent(request);
InputStream resultInputStream = result.getPayload().getRecordsInputStream();


// SDK V2
// In V2, 'selectObjectContent()' is available only on the S3AsyncClient.
// V2 handles responses using an event-based 'SelectObjectContentEventStream'.
S3AsyncClient s3ClientV2 = S3AsyncClient.builder()
    .region(Region.US_WEST_2)
    .build();

SelectObjectContentRequest request = SelectObjectContentRequest.builder()
    .bucket("bucket-name")
    .key("object-key")
    .expression("select * from S3Object")
    .expressionType(ExpressionType.SQL)
    .build();

SelectObjectContentResponseHandler handler = new SelectObjectContentResponseHandler() {
    // Implement the required abstract methods such as 'onEventStream()'.
};

CompletableFuture<Void> future = s3ClientV2.selectObjectContent(request, handler);
// The 'SelectObjectContentResponseHandler' implementation processes the results.
```

V1 的 **setBucketAcl** 到 V2 在 **PutBucketAclRequest.builder()** 上的 **acl** 方法

```
// SDK V1
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .build();

AccessControlList acl = new AccessControlList();
acl.grantPermission(GroupGrantee.AllUsers, Permission.Read);
s3ClientV1.setBucketAcl("bucket-name", acl);
```

```
// SDK V2
S3Client s3ClientV2 = S3Client.builder()
    .region(Region.US_WEST_2)
    .build();

PutBucketAclRequest request = PutBucketAclRequest.builder()
    .bucket("bucket-name")
    .acl(BucketCannedACL.PRIVATE)
    .build();

s3ClientV2.putBucketAcl(request);
```

V1 的 **setObjectAcl** 到 V2 在 **PutObjectAclRequest.builder()** 上的 **acl** 方法

```
// SDK V1
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .build();

AccessControlList acl = new AccessControlList();
acl.grantPermission(GroupGrantee.AllUsers, Permission.Read);
s3ClientV1.setObjectAcl("bucket-name", "object-key", acl);


// SDK V2
S3Client s3ClientV2 = S3Client.builder()
    .region(Region.US_WEST_2)
    .build();

PutObjectAclRequest request = PutObjectAclRequest.builder()
    .bucket("bucket-name")
    .key("object-key")
    .acl(ObjectCannedACL.PRIVATE)
    .build();

s3ClientV2.putObjectAcl(request);
```

V1 的 **initiateMultipartUpload** 到 V2 的 **createMultipartUpload**

示例迁移

```
// SDK V1
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard()
```

```
    .withRegion(Regions.US_WEST_2)
    .build();

ObjectMetadata metadata = new ObjectMetadata();
metadata.setContentType("application/zip");
metadata.addUserMetadata("mykey", "myvalue");

InitiateMultipartUploadRequest initRequest = new InitiateMultipartUploadRequest(
    "bucket-name",
    "object-key",
    metadata
);

InitiateMultipartUploadResult initResponse =
 s3ClientV1.initiateMultipartUpload(initRequest);
String uploadId = initResponse.getUploadId();


// SDK V2
// V1 uses ObjectMetadata methods, whereas V2 uses a simple Map.
S3Client s3ClientV2 = S3Client.builder()
    .region(Region.US_WEST_2)
    .build();

CreateMultipartUploadRequest createMultipartRequest =
 CreateMultipartUploadRequest.builder()
    .bucket("amzn-s3-demo-bucket")
    .key("object-key")
    .contentType("application/zip")
    .metadata(Collections.singletonMap("mykey", "myvalue"))
    .build();

CreateMultipartUploadResponse response =
 s3ClientV2.createMultipartUpload(createMultipartRequest);
String uploadId = response.uploadId();
```

实施差异

以下方法的默认 Content-Type 标头值有所不同，如下表所示。

SDK 版本	方法	默认 **Content-Type** 值
版本 1	initiateMultipartUpload	application/octet-stream
版本 2	createMultipartUpload	binary/octet-stream

V1 的 **setRegion** 到 V2 在客户端生成器上的 **region** 方法

```
// SDK V1
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard().build();
s3ClientV1.setRegion(Region.getRegion(Regions.US_WEST_2));


// SDK V2
S3Client s3ClientV2 = S3Client.builder()
    .region(Region.US_WEST_2)
    .build();
```

V1 的 **setS3ClientOptions(S3ClientOptions clientOptions)**

V2 没有在 setS3ClientOptions 方法中使用单个 S3ClientOptions 对象，而是在客户端生成器上提供方法来设置选项。

V1 的 **setBucketLoggingConfiguration** 到 V2 的 **putBucketLogging**

```
// SDK V1
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .build();

BucketLoggingConfiguration loggingConfig = new BucketLoggingConfiguration();
loggingConfig.setDestinationBucketName("log-bucket");

SetBucketLoggingConfigurationRequest request = new
 SetBucketLoggingConfigurationRequest(
    "amzn-s3-demo-source-bucket",
    loggingConfig
```

```
);

s3ClientV1.setBucketLoggingConfiguration(request);


// SDK V2
// In V2, V1's 'BucketLoggingConfiguration' is replaced by 'BucketLoggingStatus'
// and 'LoggingEnabled'.
S3Client s3ClientV2 = S3Client.builder()
    .region(Region.US_WEST_2)
    .build();

LoggingEnabled loggingEnabled = LoggingEnabled.builder()
    .targetBucket("log-bucket")
    .build();

BucketLoggingStatus loggingStatus = BucketLoggingStatus.builder()
    .loggingEnabled(loggingEnabled)
    .build();

PutBucketLoggingRequest request = PutBucketLoggingRequest.builder()
    .bucket("amzn-s3-demo-source-bucket")
    .bucketLoggingStatus(loggingStatus)
    .build();

s3ClientV2.putBucketLogging(request);
```

V1 的 **setBucketLifecycleConfiguration** 到 V2 的 **putBucketLifecycleConfiguration**

```
// SDK V1
BucketLifecycleConfiguration.Rule rule = new BucketLifecycleConfiguration.Rule()
    .withId("Archive and Delete Rule")
    .withPrefix("documents/")
    .withStatus(BucketLifecycleConfiguration.ENABLED)
    .withTransitions(Arrays.asList(
        new Transition()
            .withDays(30)
            .withStorageClass(StorageClass.StandardInfrequentAccess.toString()),
        new Transition()
            .withDays(90)
            .withStorageClass(StorageClass.Glacier.toString())
    ))
    .withExpirationInDays(365);
```

```
BucketLifecycleConfiguration configuration = new BucketLifecycleConfiguration()
    .withRules(Arrays.asList(rule));

s3ClientV1.setBucketLifecycleConfiguration("amzn-s3-demo-bucket", configuration);


// SDK V2
LifecycleRule rule = LifecycleRule.builder()
    .id("Archive and Delete Rule")
    .filter(LifecycleRuleFilter.builder()
        .prefix("documents/")
        .build())
    .status(ExpirationStatus.ENABLED)
    .transitions(Arrays.asList(
        Transition.builder()
            .days(30)
            .storageClass(TransitionStorageClass.STANDARD_IA)
            .build(),
        Transition.builder()
            .days(90)
            .storageClass(TransitionStorageClass.GLACIER)
            .build()
    ))
    .expiration(LifecycleExpiration.builder()
        .days(365)
        .build())
    .build();

PutBucketLifecycleConfigurationRequest request =
 PutBucketLifecycleConfigurationRequest.builder()
    .bucket("amzn-s3-demo-bucket")
    .lifecycleConfiguration(BucketLifecycleConfiguration.builder()
        .rules(rule)
        .build())
    .build();

s3ClientV2.putBucketLifecycleConfiguration(request);
```

V1 的 **setBucketTaggingConfiguration** 到 V2 的 **putBucketTagging**

```
// SDK V1
List<TagSet> tagsets = new ArrayList<>();
```

```
TagSet tagSet = new TagSet();
tagSet.setTag("key1", "value1");
tagSet.setTag("key2", "value2");
tagsets.add(tagSet);

BucketTaggingConfiguration bucketTaggingConfiguration = new
 BucketTaggingConfiguration();
bucketTaggingConfiguration.setTagSets(tagsets);

SetBucketTaggingConfigurationRequest request = new
 SetBucketTaggingConfigurationRequest(
    "amzn-s3-demo-bucket",
    bucketTaggingConfiguration
);

s3ClientV1.setBucketTaggingConfiguration(request);


// SDK V2
Tagging tagging = Tagging.builder()
    .tagSet(Arrays.asList(
        Tag.builder()
            .key("key1")
            .value("value1")
            .build(),
        Tag.builder()
            .key("key2")
            .value("value2")
            .build()
    ))
    .build();

PutBucketTaggingRequest request = PutBucketTaggingRequest.builder()
    .bucket("amzn-s3-demo-bucket")
    .tagging(tagging)
    .build();

s3ClientV2.putBucketTagging(request);
```

需要手动迁移的 V1 类

V1 的 **AccessControlList** 到 V2 的 **AccessControlPolicy**

```
// SDK V1
```

```
AccessControlList aclV1 = new AccessControlList();
aclV1.setOwner(new Owner("owner-id", "owner-name"));
aclV1.grantPermission(GroupGrantee.AllUsers, Permission.Read);

// SDK V2
// To migrate from V1 to V2, replace direct 'AccessControlList' modifications with an
// 'AccessControlPolicy.builder()' that contains both owner information and grants.
// Note that V2's approach requires building the complete permission set upfront rather
 than modifying permissions incrementally.
AccessControlPolicy acpV2 = AccessControlPolicy.builder()
    .owner(Owner.builder()
        .id("owner-id")
        .displayName("owner-name")
        .build())
    .grants(Arrays.asList(
         Grant.builder()
            .grantee(Grantee.builder()
                 .type(Type.GROUP)
                 .uri("http://acs.amazonaws.com/groups/global/AllUsers")
                 .build())
             .permission(Permission.READ)
             .build()
     ))
     .build();
```

V1 的 **CannedAccessControlList** 枚举到 V2 的 **BucketCannedACL** 和 **ObjectCannedACL** 枚举

```
// SDK V1
// In V1, 'CannedAccessControlList' is an enumeration of predefined ACLs.
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard().build();

// Creating a bucket.
s3ClientV1.setBucketAcl("bucket-name", CannedAccessControlList.PublicRead);

// Creating an object.
PutObjectRequest putObjectRequest = new PutObjectRequest("bucket-name", "object-key",
 file)
    .withCannedAcl(CannedAccessControlList.PublicRead);
s3ClientV1.putObject(putObjectRequest);


// SDK V2
```

```
// V2 replaces V1's 'CannedAccessControlList' with 'BucketCannedACL' for buckets and
 'ObjectCannedACL' for objects.
S3Client s3ClientV2 = S3Client.builder().build();

// Creating a bucket.
PutBucketAclRequest bucketRequest = PutBucketAclRequest.builder()
    .bucket("bucket-name")
    .acl(BucketCannedACL.PRIVATE)
    .build();
s3ClientV2.putBucketAcl(bucketRequest);

// Creating an object.
PutObjectRequest objectRequest = PutObjectRequest.builder()
    .bucket("bucket-name")
    .key("object-key")
    .acl(ObjectCannedACL.PUBLIC_READ)
    .build();
s3ClientV2.putObject(objectRequest, RequestBody.fromFile(file));
```

V1 的 **BucketNotificationConfiguration** 到 V2 的 **NotificationConfiguration**

```
//SDK V1
BucketNotificationConfiguration notificationConfig = new
 BucketNotificationConfiguration();

// Adding configurations by name
notificationConfig.addConfiguration("lambdaConfig",
    new LambdaConfiguration("arn:aws:lambda:function", "s3:ObjectCreated:"));

notificationConfig.addConfiguration("topicConfig",
    new TopicConfiguration("arn:aws:sns:topic", "s3:ObjectRemoved:"));

notificationConfig.addConfiguration("queueConfig",
    new QueueConfiguration("arn:aws:sqs:queue", "s3:ObjectRestore:*"));

s3Client.setBucketNotificationConfiguration("bucket", notificationConfig);


// SDK V2
// In V2, V1's BucketNotificationConfiguration is renamed to
 NotificationConfiguration.
// V2 contains no common abstract class for LambdaFunction/Topic/Queue configurations.
NotificationConfiguration notificationConfig = NotificationConfiguration.builder()
```

```
    .lambdaFunctionConfigurations(
        LambdaFunctionConfiguration.builder()
            .lambdaFunctionArn("arn:aws:lambda:function")
            .events(Event.valueOf("s3:ObjectCreated:"))
            .build())
    .topicConfigurations(
        TopicConfiguration.builder()
            .topicArn("arn:aws:sns:topic")
            .events(Event.valueOf("s3:ObjectRemoved:"))
            .build())
    .queueConfigurations(
        QueueConfiguration.builder()
            .queueArn("arn:aws:sqs:queue")
            .events(Event.valueOf("s3:ObjectRestore:*"))
            .build())
    .build();

s3Client.putBucketNotificationConfiguration(req -> req
    .bucket("bucket")
    .notificationConfiguration(notificationConfig));
```

V1 的 **MultiFactorAuthentication** 到 V2 在请求生成器上的 **mfa** 方法

```
// SDK V1
AmazonS3 s3ClientV1 = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.US_WEST_2)
    .build();

BucketVersioningConfiguration versioningConfig = new BucketVersioningConfiguration()
    .withStatus(BucketVersioningConfiguration.ENABLED);

// Create an MFA configuration object.
MultiFactorAuthentication mfa = new MultiFactorAuthentication(
    "arn:aws:iam::1234567890:mfa/user",
    "123456"
);

// Create the request object.
SetBucketVersioningConfigurationRequest request = new
 SetBucketVersioningConfigurationRequest(
    "bucket-name",
    versioningConfig,
    mfa
```

```
);

// Send the request.
s3ClientV1.setBucketVersioningConfiguration(request);


// SDK V2
// V2 replaces V1's MultiFactorAuthentication POJO with parameters you set on the
 request builder.
S3Client s3ClientV2 = S3Client.builder()
    .region(Region.US_WEST_2)
    .build();

PutBucketVersioningRequest request = PutBucketVersioningRequest.builder()
    .bucket("bucket-name")
    .versioningConfiguration(VersioningConfiguration.builder()
        .status(BucketVersioningStatus.ENABLED)
        .build())
    .mfa("arn:aws:iam::1234567890:mfa/user 123456")  // Single method takes both MFA
 erial number and token.
    .build();

s3ClientV2.putBucketVersioning(request);
```

将 Transfer Manager 从AWS SDK for Java 的版本 1 迁移到版本 2。

本迁移指南介绍了 Transfer Manager v1 和 S3 Transfer Manager v2 之间的主要差异，包括构造函数更改、方法映射和常见操作的代码示例。在了解这些差异之后，您可以成功迁移现有的 Transfer Manager 代码，以便充分利用 v2 中改进的性能和异步操作。

⚠ 关于 AWS SDK 迁移工具

AWS SDK for Java 提供了一个自动迁移工具，可以将大部分 v1 Transfer Manager API 迁移到 v2。但是，该迁移工具不支持几项 v1 Transfer Manager 功能。对于这些情况，您需要按照本主题中的指导手动迁移 Transfer Manager 代码。
在本指南中，迁移状态显示了迁移工具是否可以自动迁移构造函数、方法或功能：

- 支持：迁移工具可以自动转换此代码
- 不支持：您需要手动迁移代码

即使对于标记为“支持”的项目，也要检查迁移结果并进行彻底测试。Transfer Manager 迁移涉及从同步操作到异步操作的重大架构更改。

概览

S3 Transfer Manager v2 对 Transfer Manager API 进行了重大更改。S3 Transfer Manager v2 基于异步操作构建，可提供更好的性能，尤其是在使用基于 AWS CRT 的 Amazon S3 客户端时。

主要区别

- 软件包：`com.amazonaws.services.s3.transfer` → `software.amazon.awssdk.transfer.s3`
- 类名：`TransferManager` → `S3TransferManager`
- 客户端依赖项：同步 Amazon S3 客户端 → 异步 Amazon S3 客户端（`S3AsyncClient`）
- 架构：同步操作 → 使用 `CompletableFuture` 的异步操作
- 性能：通过基于 AWS CRT 的客户端支持得到增强

高级别更改

方面	V1	V2
Maven 依赖项	`aws-java-sdk-s3`	`s3-transfer-manager`
软件包	`com.amazonaws.services.s3.transfer`	`software.amazon.awssdk.transfer.s3`
主要的类	`TransferManager`	`S3TransferManager`
Amazon S3 客户端	`AmazonS3`（同步）	`S3AsyncClient` （异步）
返回类型	阻塞操作	`CompletableFuture<T>`

Maven 依赖项

V1

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>com.amazonaws</
groupId>
            <artifactId>aws-java-sdk-
bom</artifactId>
            <version>> 1.12.787[1]</
version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
<dependencies>
    <dependency>
        <groupId>com.amazonaws</gro
upId>
        <artifactId>aws-java-sdk-s3
</artifactId>
    </dependency>
</dependencies>
```

V2

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>software.
amazon.awssdk</groupId>
            <artifactId>bom</a
rtifactId>
            <version> 2.31.68[2]</version
>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
<dependencies>
    <dependency>
        <groupId>software.amazon.aw
ssdk</groupId>
        <artifactId>s3-transfer-man
ager</artifactId>
    </dependency>
    <!-- Optional: For enhanced
 performance with AWS CRT -->
    <dependency>
        <groupId>software.amazon.aw
ssdk.crt</groupId>
        <artifactId>aws-crt</artifa
ctId>
        <version> 0.38.5[3]</version>
    </dependency>
</dependencies>
```

[1] 最新版本。 [2] 最新版本。 [3] 最新版本。

客户端构造函数迁移

支持的构造函数（自动迁移）

V1 构造函数	V2 等效项	迁移状态
`new TransferManager()`	`S3TransferManager.create()`	支持
`TransferManagerBuilder.defaultTransferManager()`	`S3TransferManager.create()`	支持
`TransferManagerBuilder.standard().build()`	`S3TransferManager.builder().build()`	支持
`new TransferManager(AWSCredentials)`	`S3TransferManager.builder() .s3Client(S3AsyncClient.builder() .credentialsProvider(...).build()) .build()`	支持
`new TransferManager( AWSCredentialsProvider)`	`S3TransferManager.builder() .s3Client(S3AsyncClient.builder() .credentialsProvider(...).build()) .build()`	支持

不支持的构造函数（需要手动迁移）

V1 构造函数	V2 等效项	迁移说明
`new TransferManager(AmazonS3)`	需要手动迁移	单独创建 `S3AsyncClient`
`new TransferManager(AmazonS3, ExecutorService)`	需要手动迁移	创建 `S3AsyncClient` 并配置执行器

V1 构造函数	V2 等效项	迁移说明
new TransferManager(AmazonS3, ExecutorService, boolean)	需要手动迁移	不支持 shutDownThreadPools 参数

手动迁移示例

V1 代码：

```
AmazonS3 s3Client = AmazonS3ClientBuilder.defaultClient();
TransferManager transferManager = new TransferManager(s3Client);
```

V2 代码：

```
// Create an `S3AsyncClient` with similar configuration
S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
    .credentialsProvider(DefaultCredentialsProvider.create())
    .build();

// Provide the configured `S3AsyncClient` to the S3 transfer manager builder.
S3TransferManager transferManager = S3TransferManager.builder()
    .s3Client(s3AsyncClient)
    .build();
```

客户端方法迁移

目前，迁移工具支持基本
copy、download、upload、uploadDirectory、downloadDirectory、resumeDownload 和 resumeUpload 方法。

核心传输方法

V1 方法	V2 方法	返回类型更改	迁移状态
upload(String, String, File)	uploadFile(UploadFileRequest)	Upload → FileUpload	支持

V1 方法	V2 方法	返回类型更改	迁移状态
upload(PutObjectRequest)	upload(UploadRequest)	Upload → Upload	支持
download(String, String, File)	downloadFile(DownloadFileRequest)	Download → FileDownload	支持
download(GetObjectRequest, File)	downloadFile(DownloadFileRequest)	Download → FileDownload	支持
copy(String, String, String, String)	copy(CopyRequest)	Copy → Copy	支持
copy(CopyObjectRequest)	copy(CopyRequest)	Copy → Copy	支持
uploadDirectory(String, String, File, boolean)	uploadDirectory(UploadDirectoryRequest)	MultipleFileUpload → DirectoryUpload	支持
downloadDirectory(String, String, File)	downloadDirectory(DownloadDirectoryRequest)	MultipleFileDownload → DirectoryDownload	支持

可恢复的传输方法

V1 方法	V2 方法	迁移状态
resumeUpload(PersistableUpload)	resumeUploadFile(ResumableFileUpload)	支持
resumeDownload(PersistableDownload)	resumeDownloadFile(ResumableFileDownload)	支持

生命周期方法

V1 方法	V2 方法	迁移状态
`shutdownNow()`	`close()`	支持
`shutdownNow(boolean)`	使用 `close()` 方法手动调整代码	不支持

不支持的 V1 客户端方法

V1 方法	V2 替代方案	备注
`abortMultipartUploads(String, Date)`	使用低级别 Amazon S3 客户端	不支持
`getAmazonS3Client()`	单独保存引用	不支持；v2 中没有 getter
`getConfiguration()`	单独保存引用	不支持；v2 中没有 getter
`uploadFileList(...)`	多次调用 `uploadFile()`	不支持
使用 `TransferStateChangeListener` 参数的 `copy` 方法	使用 `TransferListener`	参见手动迁移示例
使用 `S3ProgressListener` 参数的 `download` 方法	使用 `TransferListener`	参见手动迁移示例
使用 4 个或更多参数的 `downloadDirectory` 方法		参见手动迁移示例
使用 `ObjectMetadataProvider` 参数的 `upload` 方法	在请求中设置元数据	参见手动迁移示例
使用 `*Provider` 参数的 `uploadDirectory` 方法	在请求中设置标签	参见手动迁移示例

使用 **TransferStateChangeListener** 参数的 **copy** 方法

- copy(CopyObjectRequest copyObjectRequest, AmazonS3 srcS3, TransferStateChangeListener stateChangeListener)
- copy(CopyObjectRequest copyObjectRequest, TransferStateChangeListener stateChangeListener)

```
// V1
 ---------------------------------------------------------------------------------------------
// Initialize source S3 client
AmazonS3 s3client = AmazonS3ClientBuilder.standard()
                .withRegion("us-west-2")
                .build();

// Initialize Transfer Manager
TransferManager tm = TransferManagerBuilder.standard()
                .withS3Client(srcS3)
                .build();

CopyObjectRequest copyObjectRequest = new CopyObjectRequest(
                "amzn-s3-demo-source-bucket",
                "source-key",
                "amzn-s3-demo-destination-bucket",
                "destination-key"
        );

TransferStateChangeListener stateChangeListener = new TransferStateChangeListener() {
            @Override
            public void transferStateChanged(Transfer transfer, TransferState state) {
              //Implementation of the TransferStateChangeListener
            }
        };

Copy copy = tm.copy(copyObjectRequest, srcS3, stateChangeListener);


// V2
 ---------------------------------------------------------------------------------------------
S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
                .region(Region.US_WEST_2)
                .build();
```

```
S3TransferManager transferManager = S3TransferManager.builder()
                .s3Client(s3AsyncClient)
                .build();

// Create transfer listener (equivalent to TransferStateChangeListener in v1)

TransferListener transferListener = new TransferListener() {
            @Override
            public void transferInitiated(Context.TransferInitiated context) {
               //Implementation
               System.out.println("Transfer initiated");
            }

            @Override
            public void bytesTransferred(Context.BytesTransferred context) {
                //Implementation
                System.out.println("Bytes transferred");
            }

            @Override
            public void transferComplete(Context.TransferComplete context) {
                //Implementation
                System.out.println("Transfer completed!");
            }

            @Override
            public void transferFailed(Context.TransferFailed context) {
                //Implementation
                System.out.println("Transfer failed");
            }
        };

CopyRequest copyRequest = CopyRequest.builder()
                            .copyObjectRequest(req -> req
                                .sourceBucket("amzn-s3-demo-source-bucket")
                                .sourceKey("source-key")
                                .destinationBucket("amzn-s3-demo-destination-bucket")
                                .destinationKey("destination-key")
                             )
                              .addTransferListener(transferListener) // Configure the
 transferListener into the request
                              .build();
```

```
Copy copy = transferManager.copy(copyRequest);
```

使用 S3ProgressListener 参数的 download 方法

- download(GetObjectRequest getObjectRequest, File file, S3ProgressListener progressListener)
- download(GetObjectRequest getObjectRequest, File file, S3ProgressListener progressListener, long timeoutMillis)
- download(GetObjectRequest getObjectRequest, File file, S3ProgressListener progressListener, long timeoutMillis, boolean resumeOnRetry)

```
// V1
 ---------------------------------------------------------------------------------------------
S3ProgressListener progressListener = new S3ProgressListener() {
        @Override
        public void progressChanged(com.amazonaws.event.ProgressEvent progressEvent) {
            long bytes = progressEvent.getBytesTransferred();
            ProgressEventType eventType = progressEvent.getEventType();
            // Use bytes and eventType as needed
        }

        @Override
        public void onPersistableTransfer(PersistableTransfer persistableTransfer) {

        }
    };

Download download1 = tm.download(getObjectRequest, file, progressListener);
Download download2 = tm.download(getObjectRequest, file, progressListener,
 timeoutMillis)
Download download3 = tm.download(getObjectRequest, file, progressListener,
 timeoutMillis, true)

// V2
 ---------------------------------------------------------------------------------------------
TransferListener transferListener = new TransferListener() {
    @Override
    public void transferInitiated(Context.InitializedContext context) {
        // Equivalent to ProgressEventType.TRANSFER_STARTED_EVENT
        System.out.println("Transfer initiated");
    }
```

```
    @Override
    public void bytesTransferred(Context.BytesTransferred context) {
        // Equivalent to ProgressEventType.REQUEST_BYTE_TRANSFER_EVENT
        long bytes = context.bytesTransferred();
        System.out.println("Bytes transferred: " + bytes);
    }

    @Override
    public void transferComplete(Context.TransferComplete context) {
        // Equivalent to ProgressEventType.TRANSFER_COMPLETED_EVENT
        System.out.println("Transfer completed");
    }

    @Override
    public void transferFailed(Context.TransferFailed context) {
        // Equivalent to ProgressEventType.TRANSFER_FAILED_EVENT
        System.out.println("Transfer failed: " + context.exception().getMessage());
    }
};
DownloadFileRequest downloadFileRequest =
                        DownloadFileRequest.builder()
                            .getObjectRequest(getObjectRequest)
                            .destination(file.toPath())
                            .addTransferListener(transferListener)
                            .build();

// For download1
FileDownload download = transferManager.downloadFile(downloadFileRequest);

// For download2
CompletedFileDownload completedFileDownload = download.completionFuture()
                                                      .get(timeoutMillis,
 TimeUnit.MILLISECONDS);

// For download3, the v2 SDK does not have a direct equiavalent to the `resumeOnRetry`
 method of v1.
// If a download is interrupted, you need to start a new download request.
```

使用 4 个或更多参数的 **downloadDirectory** 方法

- downloadDirectory(String bucketName, String keyPrefix, File destinationDirectory, boolean resumeOnRetry)

- downloadDirectory(String bucketName, String keyPrefix, File destinationDirectory, boolean resumeOnRetry, KeyFilter filter)
- downloadDirectory(String bucketName, String keyPrefix, File destinationDirectory, KeyFilter filter)

```
// V1
 ----------------------------------------------------------------------------------------------------
KeyFilter filter = new KeyFilter() {
            @Override
            public boolean shouldInclude(S3ObjectSummary objectSummary) {
                //Filter implementation
            }
        };
MultipleFileDownload multipleFileDownload = tm.downloadDirectory(bucketName, keyPrefix,
 destinationDirectory, filter);

// V2
 ----------------------------------------------------------------------------------------------------
// The v2 SDK does not have a direct equiavalent to the `resumeOnRetry` method of v1.
// If a download is interrupted, you need to start a new download request.
DownloadFilter filter = new DownloadFilter() {
            @Override
            public boolean test(S3Object s3Object) {
                // Filter implementation.
            }
        };

DownloadDirectoryRequest downloadDirectoryRequest =
                                DownloadDirectoryRequest.builder()
                                    .bucket(bucketName)
                                    .filter(filter)
                                    .listObjectsV2RequestTransformer(builder ->
 builder.prefix(keyPrefix))
                                    .destination(destinationDirectory.toPath())
                                    .build();

DirectoryDownload directoryDownload =
 transferManager.downloadDirectory(downloadDirectoryRequest);
```

使用 **ObjectMetadata** 参数的 **upload** 方法

- upload(String bucketName, String key, InputStream input, ObjectMetadata objectMetadata)

```
// V1
 ---------------------------------------------------------------------------------------------
ObjectMetadata metadata = new ObjectMetadata();
ObjectMetadata metadata = new ObjectMetadata();

metadata.setContentType("text/plain");        // System-defined metadata
metadata.setContentLength(22L);               // System-defined metadata
metadata.addUserMetadata("myKey", "myValue"); // User-defined metadata

PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName, key, inputStream,
 metadata);

Upload upload = transferManager.upload("amzn-s3-demo-bucket", "my-key", inputStream,
 metadata);

// V2
 ---------------------------------------------------------------------------------------------
/* When you use an InputStream to upload in V2, you should specify the content length
   and use `RequestBody.fromInputStream()`.
   If you don't provide the content length, the entire stream will be buffered in
 memory.
   If you can't determine the content length, we recommend using the CRT-based S3
 client.
*/
Map<String, String> userMetadata = new HashMap<>();
userMetadata.put("x-amz-meta-myKey", "myValue");

PutObjectRequest putObjectRequest =
                        PutObjectRequest.builder()
                            .bucket("amzn-s3-demo-bucket1")
                            .key("k")
                            .contentType("text/plain") //System-defined metadata
 usually has separate methods in the builder.
                            .contentLength(22L)
                            .metadata(userMetadata) //metadata() is only for user-
defined metadata.
                            .build();
```

```
UploadRequest uploadRequest =
                        UploadRequest.builder()
                            .putObjectRequest(putObjectRequest)
                            .requestBody(AsyncRequestBody.fromInputStream(stream, 22L,
 executor))
                            .build();

transferManager.upload(uploadRequest).completionFuture().join();
```

使用 **ObjectMetadataProvider** 参数的 **uploadDirectory**

- uploadDirectory(String bucketName, String virtualDirectoryKeyPrefix, File directory, boolean includeSubdirectories, ObjectMetadataProvider metadataProvider)
- uploadDirectory(String bucketName, String virtualDirectoryKeyPrefix, File directory, boolean includeSubdirectories, ObjectMetadataProvider metadataProvider, ObjectTaggingProvider taggingProvider)
- uploadDirectory(String bucketName, String virtualDirectoryKeyPrefix, File directory, boolean includeSubdirectories, ObjectMetadataProvider metadataProvider, ObjectTaggingProvider taggingProvider, ObjectCannedAclProvider cannedAclProvider)

```
// V1
 ---------------------------------------------------------------------------------------------
tm.uploadDirectory(bucketName, virtualDirectoryKeyPrefix, directory,
 includeSubdirectories, metadataProvider)
tm.uploadDirectory(bucketName, virtualDirectoryKeyPrefix, directory,
 includeSubdirectories, metadataProvider, taggingProvider)
tm.uploadDirectory(bucketName, virtualDirectoryKeyPrefix, directory,
 includeSubdirectories, metadataProvider, taggingProvider, cannedAclProvider)

// V2
 ---------------------------------------------------------------------------------------------
UploadDirectoryRequest request = UploadDirectoryRequest.builder()
                                    .bucket(bucketName)
                                    .s3Prefix(virtualDirectoryKeyPrefix)
                                    .source(directory.toPath())
                                    .maxDepth(includeSubdirectories ? Integer.MAX_VALUE :
 1)
```

```
                                    .uploadFileRequestTransformer(builder -> {
                                        // 1.Replace `ObjectMetadataProvider`,
 `ObjectTaggingProvider`, and `ObjectCannedAclProvider` with an
                                        // `UploadFileRequestTransformer` that can
 combine the functionality of all three *Provider implementations.
                                        // 2. Convert your v1 `ObjectMetadata` to v2
 `PutObjectRequest` parameters.
                                        // 3. Convert your v1 `ObjectTagging` to v2
 `Tagging`.
                                        // 4. Convert your v1 `CannedAccessControlList`
 to v2 `ObjectCannedACL`.
                                    })
                                    .build();

DirectoryUpload directoryUpload = transferManager.uploadDirectory(request);
```

模型对象迁移

在 AWS SDK for Java 2.x 中，许多 `TransferManager` 模型对象经过重新设计，不再支持 v1 模型对象中可用的几个 getter 和 setter 方法。

在 v2 中，您可以使用 `CompletableFuture<T>` 类在传输完成时执行操作，无论传输是成功还是出现异常。如果需要，您可以使用 `join()` 方法来等待完成。

核心传输对象

V1 类	V2 类	迁移状态
`TransferManager`	`S3TransferManager`	支持
`TransferManagerBuilder`	`S3TransferManager.Builder`	支持
`Transfer`	`Transfer`	支持
`AbortableTransfer`	`Transfer`	支持（没有单独的类）
`Copy`	`Copy`	支持
`Download`	`FileDownload`	支持
`Upload`	`Upload` / `FileUpload`	支持

V1 类	V2 类	迁移状态
MultipleFileDownload	DirectoryDownload	支持
MultipleFileUpload	DirectoryUpload	支持

持久化对象

V1 类	V2 类	迁移状态
PersistableDownload	ResumableFileDownload	支持
PersistableUpload	ResumableFileUpload	支持
PersistableTransfer	ResumableTransfer	支持
PauseResult<T>	直接可恢复对象	不支持

结果对象

V1 类	V2 类	迁移状态
CopyResult	CompletedCopy	支持
UploadResult	CompletedUpload	支持

配置对象

V1 类	V2 类	迁移状态
TransferManagerConfiguration	MultipartConfiguration （在 Amazon S3 客户端上）	支持

V1 类	V2 类	迁移状态
`TransferProgress`	`TransferProgress` + `TransferProgressSnapshot`	支持
`KeyFilter`	`DownloadFilter`	支持

不支持的对象

V1 类	V2 替代方案	迁移状态
`PauseStatus`	不支持	不支持
`UploadContext`	不支持	不支持
`ObjectCannedAclProvider`	`PutObjectRequest.builder().acl()`	不支持
`ObjectMetadataProvider`	`PutObjectRequest.builder().metadata()`	不支持
`ObjectTaggingProvider`	`PutObjectRequest.builder().tagging()`	不支持
`PresignedUrlDownload`	不支持	不支持

TransferManagerBuilder 配置迁移

配置更改

您需要为 v2 传输管理器设置的配置更改取决于您使用的 S3 客户端。您可以选择基于 AWS CRT 的 S3 客户端或基于 Java 的标准 S3 异步客户端。有关差异的信息，请参阅 the section called “SDK 中的 S3 客户端” 主题。

Use the AWS CRT-based S3 client

设置	v1	v2 – 使用基于 AWS CRT 的 S3 客户端的 Transfer Manager
（获取生成器）	`TransferManagerBuilder tmBuilder = TransferManagerBuilder.standard();`	`S3TransferManager.Builder tmBuilder = S3TransferManager.builder();`
S3 客户端	`tmBuilder.withS3Client(...); tmBuilder.setS3Client(...);`	`tmBuilder.s3Client(...);`
执行程序	`tmBuilder.withExecutorFactory(...); tmBuilder.setExecutorFactory(...);`	`tmBuilder.executor(...);`
关闭线程池	`tmBuilder.withShutDownThreadPools(...); tmBuilder.setShutdownThreadPools(...);`	不支持。关闭 S3TransferManager 时，不会关闭提供的执行程序
最小上传段大小	`tmBuilder.withMinimumUploadPartSize(...); tmBuilder.setMinimumUploadPartSize(...);`	`S3AsyncClient s3 = S3AsyncClient.crtBuilder(). minimumPartSizeInBytes(...) .build(); tmBuilder.s3Client(s3);`

设置	v1	v2 – 使用基于 AWS CRT 的 S3 客户端的 Transfer Manager

分段上传阈值	`tmBuilder.withMultipartUploadThreshold(...);`
	`tmBuilder.setMultipartUploadThreshold(...);`
		`S3AsyncClient s3 = S3AsyncClient.crtBuilder().threshold InBytes(...).build();`
		
		`tmBuilder.s3Client(s3);`

最小复制段大小	`tmBuilder.withMultipartCopyPartSize(...);`
	`tmBuilder.setMultipartCopyPartSize(...);`
		`S3AsyncClient s3 = S3AsyncClient.crtBuilder().minimumPartSizeInBytes(...).build();`
		
		`tmBuilder.s3Client(s3);`

分段复制阈值	`tmBuilder.withMultipartCopyThreshold(...);`
	`tmBuilder.setMultipartCopyThreshold(...);`
		`S3AsyncClient s3 = S3AsyncClient.crtBuilder().threshold InBytes(...).build();`
		
		`tmBuilder.s3Client(s3);`

设置	v1	v2 – 使用基于 AWS CRT 的 S3 客户端的 Transfer Manager

禁用并行下载	`tmBuilder.withDisableParallelDownloads(...);`
	`tmBuilder.setDisableParallelDownloads(...);`
		通过将默认禁用分段的基于 Java 的标准 S3 客户端传递给 Transfer Manager 来禁用并行下载。
		`S3AsyncClient s3 = S3AsyncClient.builder().build();`
		`tmBuilder.s3Client(s3);`

始终计算分段 md5	`tmBuilder.withAlwaysCalculateMultipartMd5(...);` `tmBuilder.setAlwaysCalculateMultipartMd5(...);`	不支持。

Use Java-based S3 async client

设置	v1	v2 - 使用基于 Java 的 S3 异步客户端的 Transfer Manager
（获取生成器）	`TransferManagerBuilder tmBuilder = TransferManagerBuilder.standard();`	`S3TransferManager.Builder tmBuilder = S3TransferManager.builder();`
S3 客户端	`tmBuilder.withS3Client(...);`	`tmBuilder.s3Client(...);`

设置	v1	v2 - 使用基于 Java 的 S3 异步客户端的 Transfer Manager
	`tmBuilder.setS3Client(...);`	
执行程序	`tmBuilder.withExecutorFactory(...);` `tmBuilder.setExecutorFactory(...);`	`tmBuilder.executor(...);`
关闭线程池	`tmBuilder.withShutDownThreadPools(...);` `tmBuilder.setShutdownThreadPools(...);`	不支持。关闭 S3TransferManager 时，不会关闭提供的执行程序

最小上传段大小	`tmBuilder.withMinimumUploadPartSize(...);`
	`tmBuilder.setMinimumUploadPartSize(...);`
		`S3AsyncClient s3 = S3AsyncClient.builder()`
		`.multipartConfiguration(cfg -> cfg.minimumPartSizeInBytes(...)).build();`
		
		`tmBuilder.s3Client(s3);`

设置	v1	v2 - 使用基于 Java 的 S3 异步客户端的 Transfer Manager

分段上传阈值	`tmBuilder.withMultipartUploadThreshold(...);`
	`tmBuilder.setMultipartUploadThreshold(...);`
		`S3AsyncClient s3 = S3AsyncClient.builder()`
		`.multipartConfiguration(cfg ->`
		`cfg.thresholdInBytes(...)).build();`
		
		`tmBuilder.s3Client(s3);`

最小复制段大小	`tmBuilder.withMultipartCopyPartSize(...);`
	`tmBuilder.setMultipartCopyPartSize(...);`
		`S3AsyncClient s3 = S3AsyncClient.builder()`
		`.multipartConfiguration(cfg ->`
		`cfg.minimumPartSizeInBytes(...)).build();`
		
		`tmBuilder.s3Client(s3);`

分段复制阈值	`tmBuilder.withMultipartCopyThreshold(...);`
	`tmBuilder.setMultipartCopyThreshold(...);`
		`S3AsyncClient s3 = S3AsyncClient.builder()`
		`.multipartConfiguration(cfg ->`
		`cfg.thresholdInBytes(...)).build();`
		
		`tmBuilder.s3Client(s3);`

设置	v1	v2 - 使用基于 Java 的 S3 异步客户端的 Transfer Manager

禁用并行下载	`tmBuilder.withDisableParallelDownloads(...);`
	`tmBuilder.setDisableParallelDownloads(...);`
		通过将默认禁用分段的基于 Java 的标准 S3 客户端传递给 Transfer Manager 来禁用并行下载。
		`S3AsyncClient s3 = S3AsyncClient.builder().build();`
		`tmBuilder.s3Client(s3);`

始终计算分段 md5	`tmBuilder.withAlwaysCalculateMultipartMd5(...);` `tmBuilder.setAlwaysCalculateMultipartMd5(...);`	不支持。

行为更改

异步操作

V1（阻塞）：

```
Upload upload = transferManager.upload("amzn-s3-demo-bucket", "key", file);
upload.waitForCompletion(); // Blocks until complete
```

V2（异步）：

```
FileUpload upload = transferManager.uploadFile(UploadFileRequest.builder()
    .putObjectRequest(PutObjectRequest.builder()
        .bucket("amzn-s3-demo-bucket")
        .key("key")
        .build())
```

```
    .source(file)
    .build());

CompletedFileUpload result = upload.completionFuture().join(); // Blocks until complete
// Or handle asynchronously:
upload.completionFuture().thenAccept(result -> {
    System.out.println("Upload completed: " + result.response().eTag());
});
```

错误处理

V1：如果任何子请求失败，则目录传输将完全失败。

V2：即使某些子请求失败，目录传输也会成功完成。明确检查错误：

```
DirectoryUpload directoryUpload = transferManager.uploadDirectory(request);
CompletedDirectoryUpload result = directoryUpload.completionFuture().join();

// Check for failed transfers
if (!result.failedTransfers().isEmpty()) {
    System.out.println("Some uploads failed:");
    result.failedTransfers().forEach(failed ->
        System.out.println("Failed: " + failed.exception().getMessage()));
}
```

通过字节范围提取进行并行下载

在 v2 SDK 中启用自动并行传输功能后，S3 Transfer Manager 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，使用 v1 Transfer Manager 时，对象最初的上传方式确实很重要。v1 Transfer Manager 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 v1 Transfer Manager 无法通过使用子请求来加速下载过程。

从版本 1 升级到版本 2 时解析 Amazon S3 URI 发生的更改

本主题详细说明了 Amazon S3 URI 从版本 1（v1）到版本 2（v2）发生的更改。

高级别更改

在 v1 中，要开始解析 S3 URI，可使用构造函数对 `AmazonS3URI` 进行实例化。在 v2 中，在 `S3Utilities` 的实例上调用 `parseUri()`，以便返回 `S3URI`。

更改	v1	v2
Maven 依赖项	<dependencyManagement> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> *1.12.587*[1]</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>s3</artifactId> </dependency> </dependencies>	<dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> *2.27.21*[2]</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>s3</artifactId> </dependency> </dependencies>
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 更改

行为	v1	v2

解析 S3 URI。	`URI uri = URI.create( "https://s3.amazonaws.com");`
	
	`AmazonS3Uri s3Uri = new AmazonS3URI(uri, false);`
		`S3Client s3Client = S3Client.create();`
		`S3Utilities s3Utilities = s3Client.utilities();`
		
		`S3Uri s3Uri = s3Utilities.parseUri(uri);`

从 S3 URI 检索存储桶名称。	`String bucket = s3Uri.getBucket();`	`Optional<String> bucket = s3Uri.bucket();`
检索密钥。	`String key = s3Uri.getKey();`	`Optional<String> key = s3Uri.key();`

检索区域。	`String region = s3Uri.getRegion();`	`Optional<Region> region = s3Uri.region();`
		
		`String region;`
		`if (s3Uri.region().isPresent()) {`
		`region = s3Uri.region().get().id();`
		`}`

检索 S3 URI 是否为路径风格。	`boolean isPathStyle = s3Uri.isPathStyle();`	`boolean isPathStyle = s3Uri.isPathStyle();`
检索版本 ID。	`String versionId = s3Uri.getVersionId();`	`Optional<String> versionId =`

行为	v1	v2
		`s3Uri.firstMatchingRawQueryParameter("versionId");`
检索查询参数。	不适用	`Map<String, List<String>> queryParams = s3Uri.rawQueryParameters();`

行为更改

URL 编码

v1 提供了传入标志的选项，用于指定 URI 是否应进行 URL 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键或查询参数，则必须对其进行 URL 编码。例如，您需要将空格 " " 替换为 `%20`。

S3 事件通知 API 从版本 1 到版本 2 的更改

本主题详细说明了 S3 事件通知 API 从AWS SDK for Java 版本 1.x（v1）到版本 2（v2）的更改。

高级别更改

结构变化

V1 对 `EventNotificationRecord` 类型及其属性使用静态内部类，而 v2 对 `EventNotificationRecord` 类型使用单独的公共类。

命名约定更改

在 v1 中，属性类名称包括后缀 Entity，而 v2 为了更简单的命名而省略了此后缀：例如，使用 eventData 而不是使用 eventDataEntity。

依赖项、软件包和类名的更改

在 v1 中，S3 事件通知 API 类会随着 S3 模块（artifactId `aws-java-sdk-s3`）一起以传递依赖方式导入。但在 v2 中，您需要在 `s3-event-notifications` 构件上添加依赖项。

更改	v1	v2
Maven 依赖项	<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>	<dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> *2.X.X*[1]</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifactId>s3- event-notifications</ artifactId> </dependency> </dependencies>
软件包名称	`com.amazonaws.serv ices.s3.event`	`software.amazon.aw ssdk.eventnotifica tions.s3.model`
类名	S3EventNotification S3EventNotification.S3Event NotificationRecord	S3EventNotification S3EventNotificationRecord GlacierEventData

更改	v1	v2
	S3EventNotification.GlacierEventDataEntity S3EventNotification.IntelligentTieringEventDataEntity S3EventNotification.LifecycleEventDataEntity S3EventNotification.ReplicationEventDataEntity S3EventNotification.RequestParametersEntity S3EventNotification.ResponseElementsEntity S3EventNotification.RestoreEventDataEntity S3EventNotification.S3BucketEntity S3EventNotification.S3Entity S3EventNotification.S3ObjectEntity S3EventNotification.TransitionEventDataEntity S3EventNotification.UserIdentityEntity	IntelligentTieringEventData LifecycleEventData ReplicationEventData RequestParameters ResponseElements RestoreEventData S3Bucket S3 S3Object TransitionEventData UserIdentity

[1] 最新版本。

API 更改

JSON 和 **S3EventNotification** 的相互转换

应用场景	v1	v2
从 JSON 字符串创建 S3EventNotification	`S3EventNotification notification = S3EventNotification.parseJson(message.body());`	`S3EventNotification notification = S3EventNotification.fromJson(message.body());`
将 S3EventNotification 转换为 JSON 字符串	`String json = notification.toJson();`	`String json = notification.toJson();`

S3EventNotification 的访问属性

应用场景	v1	v2
从通知中检索记录	`List<S3EventNotification.S3EventNotificationRecord> records = notifcation.getRecords();`	`List<S3EventNotificationRecord> records = notification.getRecords();`
从记录列表中检索记录	`S3EventNotification.S3EventNotificationRecord record = records.stream().findAny().get();`	`S3EventNotificationRecord record = records.stream().findAny().get();`
检索 Glacier 事件数据	`S3EventNotification.GlacierEventDataEntity glacierEventData = record.getGlacierEventData();`	`GlacierEventData glacierEventData = record.getGlacierEventData();`

应用场景	v1	v2
从 Glacier 事件中检索还原事件数据	`S3EventNotification.RestoreEventDataEntity restoreEventData = glacierEventData.getRestoreEventDataEntity();`	`RestoreEventData restoreEventData = glacierEventData.getRestoreEventData();`
检索请求参数	`S3EventNotification.RequestParametersEntity requestParameters = record.getRequestParameters();`	`RequestParameters requestParameters = record.getRequestParameters();`
检索智能分层事件数据	`S3EventNotification.IntelligentTieringEventDataEntity tieringEventData = record.getIntelligentTieringEventData();`	`IntelligentTieringEventData intelligentTieringEventData = record.getIntelligentTieringEventData();`
检索生命周期事件数据	`S3EventNotification.LifecycleEventDataEntity lifecycleEventData = record.getLifecycleEventData();`	`LifecycleEventData lifecycleEventData = record.getLifecycleEventData();`
以枚举形式检索事件名称	`S3Event eventNameAsEnum = record.getEventNameAsEnum();`	`//getEventNameAsEnum does not exist; use 'getEventName()' String eventName = record.getEventName();`

应用场景	v1	v2
检索复制事件数据	`S3EventNotification.ReplicationEventDataEntity replicationEntity = record.getReplicationEventDataEntity();`	`ReplicationEventData replicationEventData = record.getReplicationEventData();`
检索 S3 存储桶和对象信息	`S3EventNotification.S3Entity s3 = record.getS3();`	`S3 s3 = record.getS3();`
检索用户身份信息	`S3EventNotification.UserIdentityEntity userIdentity = record.getUserIdentity();`	`UserIdentity userIdentity = record.getUserIdentity();`
检索响应元素	`S3EventNotification.ResponseElementsEntity responseElements = record.getResponseElements();`	`ResponseElements responseElements = record.getResponseElements();`

迁移 **`aws-lambda-java-events`** 库版本

如果您使用 aws-lambda-java-events 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。最新版本从 S3 事件通知 API 中移除了AWS SDK for Java 1.x 上的所有依赖项。

若要详细了解 `aws-lambda-java-events` 库和适用于 Java 的 SDK 2.x 在处理 S3 事件通知方面的差异，请参阅the section called “S3 事件通知处理选项”。

配置文件更改

AWS SDK for Java 2.x 解析 `~/.aws/config` 和 `~/.aws/credentials` 中的配置文件定义，以便更贴近地模拟 AWS CLI 解析文件的方式。

适用于 Java 的 SDK 2.x：

- 通过按顺序检查 $HOME、$USERPROFILE（仅 Windows）、$HOMEDRIVE、$HOMEPATH（仅 Windows），然后检查 user.home 系统属性，来解析路径开头处的 ~/ 或 ~（后跟文件系统的默认路径分隔符）。
- 查找 AWS_SHARED_CREDENTIALS_FILE 环境变量，而不是查找 AWS_CREDENTIAL_PROFILES_FILE。
- 在配置文件中，如果配置项名称开头未包含单词 profile，则静默忽略该配置项定义。
- 静默忽略不包含字母数字、下划线或连接号字符的配置项定义（为配置文件移除了开头的 profile 单词之后）。
- 合并同一文件中重复的配置项定义的设置。
- 合并配置文件和凭证文件中重复的配置项定义的设置。
- 如果在同一个文件中找到 [profile foo] 和 [foo]，则不合并设置。
- 如果在配置文件中找到 [profile foo] 和 [foo]，则使用 [profile foo] 中的设置。
- 使用同一个文件和配置文件中最后一个重复设置的值。
- 认识用于定义注释的 ; 和 #。
- 认识配置项定义中用于定义注释的 ; 和 #，即使这些字符与右括号相邻也要认出来。
- 只有当 ; 和 # 前面有空格时，才在设置值中将其视为注释。
- 只有当 ; 和 # 前面没有空格时，才在设置值中将这些符号及后面的所有内容视为注释。
- 将基于角色的凭证视为优先级最高的凭证。如果用户指定了 role_arn 属性，2.x SDK 将始终使用基于角色的凭证。
- 将基于会话的凭证视为优先级第二高的凭证。如果未使用基于角色的凭证且用户指定了 aws_access_key_id 和 aws_session_token 属性，则 2.x SDK 将始终使用基于会话的凭证。
- 如果未使用基于角色的凭证和基于会话的凭证且用户指定了 aws_access_key_id 属性，则使用基本凭证。

环境变量和系统属性更改

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_CREDENTIAL_PROFILES_FILE		AWS_SHARED_CREDENTIALS_FILE	aws.sharedCredentialsFile
AWS_PROFILE	aws.profile	AWS_PROFILE	aws.profile
AWS_EC2_METADATA_DISABLED	com.amazonaws.sdk.disableEc2Metadata	AWS_EC2_METADATA_DISABLED	aws.disableEc2Metadata
	com.amazonaws.sdk.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
	com.amazonaws.sdk.enableThrottledRetry	不支持	不支持
	com.amazonaws.regions.RegionUtils.fileOverride	不支持（请求功能）	不支持（请求功能）
	com.amazonaws.regions.RegionUtils.disableRemote	不支持（请求功能）	不支持（请求功能）
	com.amazonaws.services.s3.disableImplicitGlobalClients	不支持（请求功能）	不支持（请求功能）
	com.amazonaws.sdk.enableInRegionOptimizedMode	不支持（请求功能）	不支持（请求功能）

Waiter 从版本 1 到版本 2 的变化

本主题详细介绍了 Waiter 的功能从版本 1（v1）到版本 2（v2）的变化。

下表展示了 DynamoDB Waiter 的具体差异。其他服务的 Waiter 也遵循同样的模式。

高级别更改

Waiter 类与服务位于相同的 Maven 构件中。

更改	v1	v2
Maven 依赖项	<dependencyManagement> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> *1.12.680*[1]</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>dynamodb</artif actId> </dependency> </dependencies>	<dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> *2.27.10*[2]</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>dynamodb</artif actId> </dependency> </dependencies>
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	• 同步：DynamoDbWaiter • 异步：DynamoDbAsyncWaiter

[1] 最新版本。[2] 最新版本。

API 更改

更改	v1	v2

创建 Waiter	`AmazonDynamoDB client = AmazonDynamoDBClientBuilder .standard().build(); AmazonDynamoDBWaiters waiter = client.waiters();`	同步：
		`DynamoDbClient client = DynamoDbClient.create(); DynamoDbWaiter waiter = client.waiter();`
		异步：
		`DynamoDbAsyncClient asyncClient = DynamoDbAsyncClient.create(); DynamoDbAsyncWaiter waiter = asyncClient.waiter();`

等待表完成创建	同步： `waiter.tableExists() .run(new WaiterParameters<>( new DescribeTableRequest(tableName)));`	同步： `WaiterResponse<DescribeTableResponse> waiterResponse = waiter.waitUntilTableExists( r -> r.tableName("myTable"));`

更改	v1	v2

	异步：
	
	`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();`
		`waiterResponse.match`
		`ed().response()`
		`       .ifPresen`
		`t(System.out::prin`
		`tln);`
		
		异步：
		
		`waiter.waitUntilTa`
		`bleExists(r ->`
		` r.tableName(tableN`
		`ame))`
		`           .whenComp`
		`lete((r, t) -> {`
		`               if (t !=`
		` null) {`
		
		` t.printStackTrace();`
		`               } else {`
		
		` System.out.println(`
		
		`    "Table creation`
		` succeeded");`
		`               }`
		`           }).join();`

配置更改

更改	v1	v2

轮询策略（最大尝试次数和固定延迟）	`MaxAttemptsRetryStrategy maxAttemptsRetryStrategy = new MaxAttemptsRetryStrategy(10);`
	
	`FixedDelayStrategy fixedDelayStrategy = new FixedDelayStrategy(3);`
	
	`PollingStrategy pollingStrategy = new PollingStrategy(maxAttemptsRetryStrategy, fixedDelayStrategy);`
	
	`waiter.tableExists().run( new WaiterParameters<>( new DescribeTableRequest(tableName)), pollingStrategy);`
		`FixedDelayBackoffStrategy fixedDelayBackoffStrategy = FixedDelayBackoffStrategy .create(Duration.ofSeconds(3));`
		
		`waiter.waitUntilTableExists(r -> r.tableName(tableName), c -> c.maxAttempts(10) .backoffStrategy(fixedDelayBackoffStrategy));`

EC2 元数据实用程序从版本 1 到版本 2 的更改

本主题详细介绍了适用于 Java 的 SDK 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	<dependencyManagement> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> 1.12.587[1]</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-co re</artifactId> </dependency> </dependencies>	<dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> 2.27.21[2]</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>imds</artifactId> </dependency> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>apache-client [3]</ artifactId> </dependency> </dependencies>

更改	v1	v2
软件包名称	`com.amazonaws.util`	`software.amazon.awssdk.imds`

实例化方法	使用静态实用程序方法；不进行实例化：
	`String localHostName = EC2MetadataUtils.getLocalHostName();`
		使用静态工厂方法：
		`Ec2MetadataClient client = Ec2MetadataClient.create();`
		或者使用生成器方法：
		`Ec2MetadataClient client = Ec2MetadataClient.builder() .endpointMode(EndpointMode.IPV6) .build();`

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 需要实现同步元数据客户端的`SdkHttpClient`接口或异步元数据客户端的`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，您需要在 v2 中将 EC2 资源路径指定为参数。下表显示了不同的方法。

v1	v2

`String userMetaData = EC2Metadata.getUserData();`	`Ec2MetadataClient client = Ec2MetadataClient.create();`
	`Ec2MetadataResponse response = client.get("/latest/user-data");`
	`String userMetaData = response.asString();`

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

开启后 EC2，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。

使用案例	v1	v2
检索区域	`InstanceInfo instanceInfo = EC2MetadataUtils.getInstanceInfo();`	`Ec2MetadataResponse response = client.get("/latest/dynamic/instance-identity/document");`

使用案例	v1	v2
	`String region = instanceInfo.getRegion();`	`Document instanceInfo = response.asDocument(); String region = instanceInfo.asMap().get("region").asString();`
检索实例 ID	`InstanceInfo instanceInfo = EC2MetadataUtils.getInstanceInfo(); String instanceId = instanceInfo.instanceId;`	`Ec2MetadataResponse response = client.get("/latest/dynamic/instance-identity/document"); Document instanceInfo = response.asDocument(); String instanceId = instanceInfo.asMap().get("instanceId").asString();`
检索实例类型	`InstanceInfo instanceInfo = EC2MetadataUtils.getInstanceInfo(); String instanceType = instanceInfo.instanceType();`	`Ec2MetadataResponse response = client.get("/latest/dynamic/instance-identity/document"); Document instanceInfo = response.asDocument(); String instanceType = instanceInfo.asMap().get("instanceType").asString();`

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：com.amazonaws.sdk.ec2MetadataServiceEndpointOverride	客户端生成器配置方法：endpoint(...)
环境变量：AWS_EC2_METADATA_SERVICE_ENDPOINT	系统属性：aws.ec2MetadataServiceEndpoint
默认值：http://169.254.169.254	Config 文件：~.aws/config ，及 ec2_metadata_service_endpoint 设置
	与已解析的 endpoint-mode 相关联的值
	默认值：http://169.254.169.254

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使用或的默认端点值。 IPv4 IPv6端点模式不适用于 v1。使用的默认值为http://169.254.169.254和 IPv4 fo http://[fd00:ec2::254] r IPv6。

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：endpointMode(...)	`Ec2MetadataClient client = Ec2MetadataClient.builder().endpointMode(EndpointMode.IPV4).build();`	EndpointMode.IPV4 ，EndpointMode.IPV6
系统属性	aws.ec2MetadataServiceEndpointMode	IPv4、IPv6（大小写没有影响）
Config 文件：~.aws/config	ec2_metadata_service_endpoint 设置	IPv4、IPv6（大小写没有影响）
未在前面的方法中指定	IPv4 被使用	

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，并且您尚未在代码中配置元数据客户端，则 SDK 将使用 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 SDKs 如何与 IMDS 配合使用。

v1	v2
IMDSv2 默认使用	始终使用 IMDSv2
尝试为每个请求获取会话令牌，IMDSv1 如果无法获取会话令牌，则回退到会话令牌	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

适用于 Java 的 SDK 2.x 仅支持 IMDSv2 ，不能回退到。 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 `retryPolicy(...)` 配置
HTTP	连接超时可通过 `AWS_METADATA_SERVICE_TIMEOUT` 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 `httpClient(...)` 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 到版本 2 的更改

本主题详细介绍了 Amazon CloudFront 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	<dependencyManagement> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> *1.12.587*[1]</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>cloudfront</art ifactId> </dependency> </dependencies>	<dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> *2.27.21*[2]</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>cloudfront</art ifactId> </dependency> </dependencies>
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 更改

行为	v1	v2
构建标准请求	参数直接传递给 API。	`CannedSignerRequest cannedRequest = CannedSignerRequest.builder() .resourceUrl(resourceUrl) .privateKey(privateKey) .keyPairId(keyPairId) .expirationDate(expirationDate) .build();`
构建自定义请求	参数直接传递给 API。	`CustomSignerRequest customRequest = CustomSignerRequest.builder()`

行为	v1	v2
		.resourceUrl(resou rceUrl) .resourceUrlPatter n(resourceUrlPattern) .privateKey(keyFile) .keyPairId(keyPairId) .expirationDate(ex pirationDate) .activeDate(active Date) .ipRange(ipRange) .build();
生成签名 URL（标准）	String signedUrl = CloudFrontUrlSigne r.getSignedURLWith CannedPolicy(resourceUrl, keyPairId, privateKe y, expirationDate);	CloudFrontUtilities cloudFrontUtilities = CloudFrontUtilitie s.create(); SignedUrl signedUrl = cloudFrontUtilitie s.getSignedUrlWith CannedPolicy(canne dRequest); String url = signedUrl .url();

行为	v1	v2

生成签名 Cookie（自定义）	`CookiesForCustomPolicy cookies = CloudFrontCookieSigner.getCookiesForCustomPolicy( resourceUrl, privateKey, keyPairId, expirationDate, activeDate, ipRange);`	`CloudFrontUtilities cloudFrontUtilities = CloudFrontUtilities.create();`
		
		`CookiesForCustomPolicy cookies = cloudFrontUtilities.getCookiesForCustomPolicy(customRequest);`

v2 中重构的 Cookie 标头

在 Java v1 中，Java SDK 以 `Map.Entry<String, String>` 形式提供 Cookie 标头。

```
Map.Entry<String, String> signatureMap = cookies.getSignature();
String signatureKey = signatureMap.getKey(); // "CloudFront-Signature"
String signatureValue = signatureMap.getValue(); // "[SIGNATURE_VALUE]"
```

Java v2 SDK 以单个 `String` 形式提供整个标头。

```
String signatureHeaderValue = cookies.signatureHeaderValue(); // "CloudFront-
Signature=[SIGNATURE_VALUE]"
```

IAM 策略生成器 API 从版本 1 到版本 2 的变化

本主题详细说明了 IAM 策略生成器 API 从版本 1（v1）到版本 2（v2）发生的变化。

高级别更改

更改	v1	v2
Maven 依赖项	`<dependencyManagement> <dependencies> <dependency>`	`<dependencyManagement> <dependencies> <dependency>`

更改	v1	v2
	<groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> *1.12.587*[1]</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-co re</artifactId> </dependency> </dependencies>	<groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> *2.27.21*[2]</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>iam-policy-buil der</artifactId> </dependency> </dependencies>
软件包名称	`com.amazonaws.auth.policy`	`software.amazon.awssdk.policybuilder.iam`

更改	v1	v2
类名	策略 语句 • Statement.Effect • IdentiyManagementActions • 资源： • 主体 • Condition	IamPolicy IamStatement • IamEffect • IamAction • IamResource • IamPrincipal • IamCondition • IamConditionOperator • IamConditionKey

[1] 最新版本。[2] 最新版本。

API 更改

设置	v1	v2
实例化策略	`Policy policy = new Policy();`	`IamPolicy.Builder policyBuilder = IamPolicy.builder(); ... IamPolicy policy = policyBuilder.build();`
设置 ID	`policy.withtId(...); policy.setId(...);`	`policyBuilder.id(...);`
设置版本	不适用 - 使用默认版本 2012-10-17	`policyBuilder.version(...);`
创建语句	`Statement statement =`	`IamStatement statement =`

设置	v1	v2
	`new Statement (Effect.Allow) .withActions(...) .withConditions(...) .withId(...) .withPrincipals(...) .withResources(...);`	`IamStatement.Builder() .effect(IamEffect.ALLOW) .actions(...) .notActions(...) .conditions(...) .sid(...) .principals(...) .notPrincipals(...) .resources(...) .notResources(...) .build()`
设置语句	`policy.withStatements(statement); policy.setStatements(statement);`	`policyBuilder.addStatement(statement);`

构建语句时的差异

操作

v1

v1 SDK 为服务操作使用 enum 类型，表示策略语句中的 *Action* 元素。以下 enum 类型是一些示例。

- IdentityManagementActions
- DynamoDBv2Actions
- SQSActions

以下示例显示 SQSActions 的 SendMessage 常量。

```
Action action = SQSActions.SendMessage;
```

在 v1 中，您无法为语句指定 *NotAction* 元素。

v2

在 v2 中，所有操作均由 IamAction 接口统一表示。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");
```

如以下代码所示，使用 v2 时，您可为语句指定 *NotAction*。

```
IamAction action = IamAction.create("sqs:SendMessage");
IamStatement.builder().addNotAction(action);
```

条件

v1

为了表示语句条件，v1 SDK 使用 Condition 的子类。

- 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 IamCondition 并提供 IamConditionOperator（包含所有类型的 enums 条件）来为策略语句构建条件。

```
IamCondition condition = IamCondition.create(IamConditionOperator.STRING_NOT_LIKE,
 "key", "value");
```

资源

v1

策略语句的 *Resource* 元素用 SDK 的 Resource 类表示。在构造函数中以字符串形式提供 ARN。以下子类提供了便捷的构造函数。

- S3BucketResource
- S3ObjectResource
- SQSQueueResource

在 v1 中，您可以通过调用 withIsNotType 方法为 Resource 指定 *NotResource* 元素，如以下语句所示。

```
Resource resource = new Resource("arn:aws:s3:::amzn-s3-demo-
bucket").withIsNotType(true);
```

v2

在 v2 中，您可以通过将 ARN 传递给 IamResource.create 方法来创建 *Resource* 元素。

```
IamResource resource = IamResource.create("arn:aws:s3:::amzn-s3-demo-bucket");
```

IamResource 可以设置为 *NotResource* 元素，如以下代码段所示。

```
IamResource resource = IamResource.create("arn:aws:s3:::amzn-s3-demo-bucket");
IamStatement.builder().addNotResource(resource);
```

IamResource.ALL 表示所有资源。

主体

v1

v1 SDK 提供以下 Principal 类来表示包括所有成员的主体的类型：

- AllUsers
- AllServices
- AllWebProviders
- All

您无法向语句添加 *NotPrincipal* 元素。

v2

在 v2 中，IamPrincipal.ALL 表示所有主体：

要表示其他类型主体中的所有成员，请在创建 IamPrincipal 时使用 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 Web 提供程序。
- IamPrincipal.create(IamPrincipalType.CANONICAL_USER,"*") 适用于所有规范用户。

在创建策略语句时，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);
```

从AWS SDK for Java 版本 1 升级到版本 2 时使用 DynamoDB 所发生的变化

主题

- AWS SDK for Java 版本 1 和版本 2 之间的 DynamoDB 映射 API 差异
- AWS SDK for Java 版本 1 和版本 2 之间的文档 API 差异

- 加密库迁移

AWS SDK for Java 版本 1 和版本 2 之间的 DynamoDB 映射 API 差异

AWS SDK for Java 的版本 1 和版本 2 之间的 DynamoDB 映射 API 发生了重大变化。在版本 1 中，使用 `DynamoDBMapper` 来处理 Java POJO。在版本 2 中使用 `DynamoDbEnhancedClient`，该方法具有更新的方法名称、增强的架构定义选项和改进的类型安全性。

关键差异包括：

- 新的方法名称（例如 `getItem`，而不使用 `load`）
- 显式表架构创建
- 同步和异步操作的内置支持
- 空字符串和配置的处理方式发生了变化

本节介绍映射 API 更改、注释差异、配置更新和迁移指导，以协助您从 v1 `DynamoDBMapper` 转变到 v2 `DynamoDbEnhancedClient`。

目录

- 适用于 Java 的 SDK 版本 1 升级到版本 2 时，映射库发生的高级别更改
  - 导入依赖项差异
- 适用于 Java 的 SDK 版本 1 和版本 2 之间的 DynamoDB 映射 API 更改
  - 创建客户端
  - 建立到 DynamoDB 表/索引的映射
  - 表操作
  - 映射类和属性
    - bean 注释
    - V2 附加注释
  - 配置
    - 每个操作的配置
  - 条件
  - 类型转换
    - 默认转换器
    - 为属性设置自定义转换器

    - 添加类型转换器工厂或提供程序
- 适用于 Java 的 SDK 版本 1 和版本 2 之间的字符串处理差异
- 适用于 Java 的 SDK 版本 1 和版本 2 之间的乐观锁差异
- 适用于 Java 的 SDK 版本 1 和版本 2 之间的 fluent setter 方法差异

适用于 Java 的 SDK 版本 1 升级到版本 2 时，映射库发生的高级别更改

在 V1 和 V2 中，每个库中映射客户端的名称不同：

- V1 - DynamoDBMapper
- V2 - DynamoDB 增强型客户端

您与这两个库的交互方式大致相同：实例化映射器/客户端，然后将 Java POJO 传递给负责从 DynamoDB 表中读取和写入这些对象的 API。这两个库还为 POJO 的类提供了注释，以指导客户端如何处理 POJO。

迁移到 V2 时的显著差异包括：

- V2 和 V1 为低级 DynamoDB 操作使用不同的方法名称。例如：

V1	V2
负载	getItem
save	putItem
batchLoad	batchGetItem

- V2 提供了多种定义表架构和将 POJO 映射到表的方法。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，您需要在最初几个步骤中明确地创建表架构；而在 V1 中，根据需要从注释类中推断出表架构。
- V2 在增强型客户端 API 中包括文档 API 客户端，而 V1 则使用单独的 API。
- 在 V2 中，所有 API 均提供同步和异步版本。

有关 V2 增强型客户端的更多详细信息，请参阅本指南中的 DynamoDB 映射部分。

导入依赖项差异

V1	V2
<dependencyManagement> <dependencies> <dependency> <groupId>com.amazonaws</gro upId> <artifactId>aws-java-sdk-bom</ artifactId> <version> *1.X.X*</version> <type>pom</type> <scope>import</scope> </dependency> </dependencies> </dependencyManagement> <dependencies> <dependency> <groupId>com.amazonaws</groupId> <artifactId>aws-java-sdk-dy namodb</artifactId> </dependency> </dependencies>	<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>

[*] 最新版本。

在 V1 中，单个依赖项包括低级 DynamoDB API 和映射/文档 API，而在 V2 中，您可以使用 dynamodb-enhanced 构件依赖项来访问映射/文档 API。dynamodb-enhanced 模块包含低级 dynamodb 模块的传递依赖项。

适用于 Java 的 SDK 版本 1 和版本 2 之间的 DynamoDB 映射 API 更改

创建客户端

应用场景	V1	V2
正常实例化	AmazonDynamoDB standardClient =	DynamoDbClient standardClient =

应用场景	V1	V2

	`AmazonDynamoDBClientBuilder.standard()`
	`.withCredentials(credentialsProvider)`
	`.withRegion(Regions.US_EAST_1)`
	`.build();`
	`DynamoDBMapper mapper = new DynamoDBMapper(standardClient);`
		`DynamoDbClient.builder()`
		`.credentialsProvider(ProfileCredentialsProvider.create())`
		`.region(Region.US_EAST_1)`
		`.build();`
		`DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()`
		`.dynamoDbClient(standardClient)`
		`.build();`

最小实例化	`AmazonDynamoDB standardClient = AmazonDynamoDBClientBuilder.standard();` `DynamoDBMapper mapper = new DynamoDBMapper(standardClient);`	`DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.create();`

带属性转换器*	`DynamoDBMapper mapper = new DynamoDBMapper(standardClient,`
	
	`attributeTransformerInstance);`
		`DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()`
		`.dynamoDbClient(standardClient)`
		`.extensions(extensionAInstance, extensionBInstance)`
		`.build();`

*V2 中的扩展与 V1 中的属性转换器大致对应。the section called “使用扩展自定义操作” 部分包含有关 V2 中扩展的更多信息。

建立到 DynamoDB 表/索引的映射

在 V1 中，通过 bean 注释指定 DynamoDB 表名。在 V2 中，使用一种工厂方法 `table()` 来生成代表远程 DynamoDB 表的 `DynamoDbTable` 实例。`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	`@DynamoDBTable(tableName ="Customer")`
	`public class Customer {`
	`  ...`
	`}`
		`DynamoDbTable<Customer> customerTable = enhancedClient.table("Customer",`
		`  TableSchema.fromBean(Customer.class));`

映射到 DynamoDB 二级索引	1. 定义一个表示索引的 POJO 类。
	• 给类加上 `@DynamoDBTable` 这个注解，在注解中提供包含索引的表的名称。
	• 使用 `@DynamoDBIndexHashKey` 为属性添加注释，并可选择使用 `@DynamoDBIndexRangeKey` 为属性添加注释。
	2. 创建查询表达式。
	3. 使用代表该索引的 POJO 类的引用进行查询。例如
	`mapper.query(IdEmailIndex.class, queryExpression)`
	其中 `IdEmailIndex` 是索引的映射类。
		1. 使用 `@DynamoDbSecondaryPartitionKey`（对于 GSI）和 `@DynamoDbSecondarySortKey`（对于 GSI 或 LSI）对 POJO 类的属性添加注释。例如，
		`@DynamoDbSecondarySortKey(indexNames = "IdEmailIndex")`
		`public String getEmail() { return this.email; }`
		2. 检索对索引的引用。例如，
		`DynamoDbIndex<Customer> customerIndex = customerTable.index("IdEmailIndex");`
		3. 查询索引。

应用场景	V1	V2
	《DynamoDB 开发人员指南》中讨论 V1 query 方法的部分展示了一个完整的示例。	本指南中的 ??? 部分提供了更多信息。

表操作

本节介绍大多数标准使用案例的 V1 和 V2 之间存在差异的操作 API。

在 V2 中，所有涉及单个表的操作都是在 DynamoDbTable 实例上调用，而不是在增强型客户端上调用。增强型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 或称为某个具体类型（例如 customer1）。对于 V2 示例，名为 table 的实例是先前调用 enhancedClient.table() 的结果，该调用返回对 DynamoDbTable 实例的引用。

请注意，即使未显示，也可以使用 fluent 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，表操作（在下方）包含一些常用的表单，并未涵盖所有重载表单。例如，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)
```

表操作（在下方）显示了常用的表单：

```
mapper.load(item)
mapper.load(item, config)
```

表操作

应用场景	V1	V2

将 Java POJO 写入 Dynam 表
 	
Dynam 操作：Pu m
	`mapper.save(item)`
	`mapper.save(item, config)`
	`mapper.save(item, saveExpre ssion, config)`
	
	在 V1 中，DynamoDBM apperConfig.SaveBe havior 和注释决定了将调用哪个低级 DynamoDB 方法。通常，除非在使用 SaveBehavior.CLOBB ER 和 SaveBehavior.PUT ，否则会调用 UpdateItem 。自动生成的密钥是一种特殊的使用案例，偶尔会同时使用 PutItem 和 UpdateItem 。
		`table.putItem(putItemRequest)`
		`table.putItem(item)`
		`table.putItemWithResponse(item) //` `Returns metadata.`
		
		`updateItem(updateItemRequest)`
		`table.updateItem(item)`
		`table.updateItemWithResponse(item) //` `Returns metadata.`

从 Dynam 表中将项目读取到 Java POJO
 	
Dynam 操作：Ge
	`mapper.load(item)`
	`mapper.load(item, config)`
		`table.getItem(getItemRequest)`
		`table.getItem(item)`
		`table.getItem(key)`
		`table.getItemWithResponse(key) //` `Returns POJO with metadata.`

从 Dynam 表中	`mapper.delete(item,` `deleteExpression, config)`	`table.deleteItem(deleteItemRequest)`
		`table.deleteItem(item)`
		`table.deleteItem(key)`

应用场景	V1	V2
删除项目 Dynam操作：De m		

查询Dynam表或二级索引并返回分页列表
 	
Dynam操作：Qu
	`mapper.query(Customer.class, queryExpression)`
	`mapper.query(Customer.class, queryExpression,`
	
	`mapperConfig)`
		`table.query(queryRequest)`
		`table.query(queryConditional)`
		
		使用返回的 `PageIterable.stream()` （延迟加载）进行同步响应，使用 `PagePublisher.subscribe()` 进行异步响应

查询Dynam表或二级索引并返回列表
 	
Dynam操作：Qu
	`mapper.queryPage(Customer.class, queryExpression)`
	`mapper.queryPage(Customer.class, queryExpression,`
	
	`mapperConfig)`
		`table.query(queryRequest)`
		`table.query(queryConditional)`
		
		使用返回的 `PageIterable.items()` （延迟加载）进行同步响应，使用 `PagePublisher.items.subscribe()` 进行异步响应

应用场景	V1	V2

扫描 Dynam 表或二级索引并返回分页列表
 	
Dynam 操作：Sc
	`mapper.scan(Customer.class, scanExpression)`
	`mapper.scan(Customer.class, scanExpression, mapperConfig)`
		`table.scan()`
		`table.scan(scanRequest)`
		
		使用返回的 `PageIterable.stream()` （延迟加载）进行同步响应，使用 `PagePublisher.subscribe()` 进行异步响应

扫描 Dynam 表或二级索引并返回列表
 	
Dynam 操作：Sc
	`mapper.scanPage(Customer.class, scanExpression)`
	`mapper.scanPage(Customer.class, scanExpression, mapperConfig)`
		`table.scan()`
		`table.scan(scanRequest)`
		
		使用返回的 `PageIterable.items()` （延迟加载）进行同步响应，使用 `PagePublisher.items.subscribe()` 进行异步响应

应用场景	V1	V2
批量读取多个表中的多个项目 Dynam操作：Batem	mapper.batchLoad(Arrays.asList(customer1, customer2, book1)) mapper.batchLoad(itemsToGet) // itemsToGet: Map<Class<?>, List<KeyPair>>	enhancedClient.batchGetItem(batchGetItemRequest) enhancedClient.batchGetItem(r -> r.readBatches(ReadBatch.builder(Record1.class) .mappedTableResource(mappedTable1) .addGetItem(i -> i.key(k -> k.partitionValue(0))) .build(), ReadBatch.builder(Record2.class) .mappedTableResource(mappedTable2) .addGetItem(i -> i.key(k -> k.partitionValue(0))) .build())) // Iterate over pages with lazy loading or over all items from the same table.
批量将多个项目写入多个表 Dynam操作：BateItem	mapper.batchSave(Arrays.asList(customer1, customer2, book1))	enhancedClient.batchWriteItem(batchWriteItemRequest) enhancedClient.batchWriteItem(r -> r.writeBatches(WriteBatch.builder(Record1.class) .mappedTableResource(mappedTable1) .addPutItem(item1) .build(), WriteBatch.builder(Record2.class) .mappedTableResource(mappedTable2) .addPutItem(item2) .build()))

应用场景	V1	V2

批量删除多个表中的多个项目
 	
Dynam操作：BaeItem
	`mapper.batchDelete(Arrays.asList(customer1,`
	
	`customer2,`
	
	`book1))`
		`enhancedClient.batchWriteItem(r -> r.writeBatches(`
		`WriteBatch.builder(Record1.class)`
		`.mappedTableResource(mappedTable1)`
		`.addDeleteItem(item1key)`
		`.build(),`
		`WriteBatch.builder(Record2.class)`
		`.mappedTableResource(mappedTable2)`
		`.addDeleteItem(item2key)`
		`.build()))`

批量写入/删除多个项目
 	
Dynam操作：BaeItem
	`mapper.batchWrite(Arrays.asList(customer1, book1),`
	`Arrays.asList(customer2))`
		`enhancedClient.batchWriteItem(r -> r.writeBatches(`
		`WriteBatch.builder(Record1.class)`
		`.mappedTableResource(mappedTable1)`
		`.addPutItem(item1)`
		`.build(),`
		`WriteBatch.builder(Record2.class)`
		`.mappedTableResource(mappedTable2)`
		`.addDeleteItem(item2key)`
		`.build()))`

应用场景	V1	V2
执行事务性写入 Dynam操作：TrriteIt	`mapper.transactionWrite(transactionWriteRequest)`	`enhancedClient.transactWriteItems(transasctWriteItemsRequest)`
执行事务性读取 Dynam操作：TretIten	`mapper.transactionLoad(transactionLoadRequest)`	`enhancedClient.transactGetItems(transactGetItemsRequest)`

应用场景	V1	V2

获取查询中匹配项的计数
 	
Dynam操作：使用 Selec UNT 执行 Query
	`mapper.count(Customer.class , queryExpression)`	`// Get the count from query results.`
		`PageIterable<Customer> pageIterable =`
		`customerTable.query(QueryEn hancedRequest.builder()`
		`.queryConditional(queryCond itional)`
		`.select(Select.COUNT)`
		`.build());`
		`Iterator<Page<Customer>> iterator = pageIterable.iterator();`
		`Page<Customer> page = iterator.next();`
		`int count = page.count();`
		
		`// For a more concise approach, you can chain the method calls:`
		`int count = customerTable.query(QueryEn hancedRequest.builder()`
		`.queryConditional( queryConditional)`
		`.select(Select.COUNT)`
		`.build())`
		`.iterator().next().count();`

应用场景	V1	V2

获取扫描中匹配项的计数
 	
Dynam操作：使用 Selec UNT 执行 Scan
	`mapper.count(Customer.class , scanExpression)`	`// Get the count from scan results.`
		`PageIterable<Customer> pageIterable =`
		`customerTable.scan(ScanEnha ncedRequest.builder()`
		`.filterExpression(filterExp ression)`
		`.select(Select.COUNT)`
		`.build());`
		`Iterator<Page<Customer>> iterator =`
		`pageIterable.iterator();`
		`Page<Customer> page = iterator.next();`
		`int count = page.count();`
		
		`// For a more concise approach, you can chain the method calls:`
		`int count = customerTable.scan(ScanEnha ncedRequest.builder()`
		`.filterExpression( filterExpression)`
		`.select(Select.COUNT)`
		`.build())`
		`.iterator().next().count();`

在 Dynam中创建与 POJO 类对应的表
 	
Dynam操作：Cr le
	`mapper.generateCreateTableR equest(Customer.class)`
	
	前面的语句生成一个低级别的创建表请求；用户必须在 DynamoDB 客户端中调用 `createTable` 。
		`table.createTable(createTableRequest)`
		
		`table.createTable(r -> r.provisi onedThroughput(defaultThroughput())`
		`.globalSecondaryIndices(`
		`EnhancedGlobalSecondaryInde x.builder()`
		`.indexName("gsi_1")`
		`.projection(p -> p.project ionType(ProjectionType.ALL))`
		`.provisionedThroughput(defa ultThroughput())`
		`.build()));`

应用场景	V1	V2
在 Dynam 中执行并行扫描 Dynam 操作：使用 Segmer 和 Totals ents 参数执行 Scan	`mapper.parallelScan(Customer.class, scanExpression, numTotalSegments)`	用户需要处理工作线程并为每个分段调用 scan： `table.scan(r -> r.segment(0).totalSegments(5))`
将 Amazon S3 与 Dynam 集成以存储智能 S3 链接	`mapper.createS3Link(bucket, key)` `mapper.getS3ClientCache()`	不支持，因为它将 Amazon S3 与 DynamoDB 结合在一起。

映射类和属性

在 V1 和 V2 中，都使用 bean 样式的注释将类映射到表。V2 还提供了其他方法来为特定使用案例定义架构，例如使用不可变类。

bean 注释

下表显示了 V1 和 V2 中使用的特定使用案例的等效 bean 注释。Customer 类场景用于说明参数。

V2 中的注释（以及类和枚举）遵循驼峰命名法规范，使用“DynamoDb”，而不是“DynamoDB”。

应用场景	V1	V2
将类映射到表	`@DynamoDBTable (tableName ="CustomerTable")`	`@DynamoDbBean` `@DynamoDbBean(converterProviders = {...})` 表名在调用 DynamoDbEnhancedClient#table() 方法时定义。
将类成员指定为表属性	`@DynamoDBAttribute (attributeName = "customerName")`	`@DynamoDbAttribute("customerName")`
将类成员指定为哈希键/分区键	`@DynamoDBHashKey`	`@DynamoDbPartitionKey`
将类成员指定为范围键/排序键	`@DynamoDBRangeKey`	`@DynamoDbSortKey`
将类成员指定为二级索引哈希键/分区键	`@DynamoDBIndexHashKey`	`@DynamoDbSecondaryPartitionKey`
将类成员指定为二级索引范围键/排序键	`@DynamoDBIndexRangeKey`	`@DynamoDbSecondarySortKey`

应用场景	V1	V2
映射到表时忽略此类成员	`@DynamoDBIgnore`	`@DynamoDbIgnore`
将类成员指定为自动生成的 UUID 键属性	`@DynamoDBAutoGeneratedKey`	`@DynamoDbAutoGeneratedUuid` 默认情况下，不加载提供此功能的扩展；您必须将扩展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	`@DynamoDBAutoGeneratedTimestamp`	`@DynamoDbAutoGeneratedTimestampAttribute` 默认情况下，不加载提供此功能的扩展；您必须将扩展添加到客户端生成器中。
将类成员指定为自动递增的版本属性	`@DynamoDBVersionAttribute`	`@DynamoDbVersionAttribute` 提供此功能的扩展会自动加载。
将类成员指定为需要自定义转换	`@DynamoDBTypeConverted`	`@DynamoDbConvertedBy`
将类成员指定为存储为另一个属性类型	`@DynamoDBTyped(<DynamoDBAttributeType>)`	使用 `AttributeConverter` 实现。V2 为常用 Java 类型提供许多内置转换器。您也可以实施自己的自定义 `AttributeConverter` 或 `AttributeConverterProvider`。请参阅本指南中的the section called “控制属性转换”。

应用场景	V1	V2
指定一个可以序列化为 DynamoDB 文档（JSON 风格的文档）或子文档的类	`@DynamoDBDocument`	使用增强型文档 API。请参阅以下资源： • the section called “处理 JSON 文档”本指南中的 • 发布增强型文档 API 时发表的博客文章

V2 附加注释

应用场景	V1	V2
如果 Java 值为 null，则将类成员指定为不存储为 Null 属性	不适用	`@DynamoDbIgnoreNulls`
如果所有属性都为 null，则将类成员指定为空对象	不适用	`@DynamoDbPreserveEmptyObject`
为类成员指定特殊更新操作	不适用	`@DynamoDbUpdateBehavior`
指定一个不可变类	不适用	`@DynamoDbImmutable`
将类成员指定为自动递增的计数器属性	不适用	`@DynamoDbAtomicCounter` 提供此功能的扩展会自动加载。

配置

在 V1 中，通常使用 `DynamoDBMapperConfig` 的实例来控制特定行为。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 中的默认设置	V2
	`DynamoDBMapperConfig.builder()`		
批量加载/写入重试策略	`.withBatchLoadRetryStrategy(loadRetryStrategy)` `.withBatchWriteRetryStrategy(writeRetryStrategy)`	重试失败的项目	在底层 `DynamoDBClient` 中配置重试策略。请参阅本指南中的the section called “重试”。
一致性读取	`.withConsistentReads(CONSISTENT)`	EVENTU	默认情况下，读取操作的一致性读取为 false。在请求对象上使用 `.consistentRead(true)` 覆盖。
包含编组器/解组器集合的转换架构	`.withConversionSchema(conversionSchema)` 静态实现与旧版本向后兼容。	V2_COMPATIBLE	不适用。这是一项遗留特性，指的是 DynamoDB 的最早版本（V1）存储数据类型的方式，而在增强型客户端中不再保留此行为。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	`.withObjectTableNameResolver()`
	`.withTableNameOverride()`
	`.withTableNameResolver()`
		使用注释或根据类来推断	表名在调用 `DynamoDbEnhancedClient#table()` 方法时定义。

应用场景	V1	V1 中的默认设置	V2
	静态实现与旧版本向后兼容		
分页加载策略	`.withPaginationLoadingStrategy(strategy)` 选项包括：LAZY_LOADING、EAGER_LOADING 或 ITERATION_ONLY	LAZY_L ING	• 默认为“仅迭代”。不支持其他 V1 选项。 • 您可以通过以下方式在 V2 中实施等同于预先加载的功能： `List<Customer> allItems = customerTable.scan().items().stream().collect(Collectors.toList());` • 对于延迟加载，您必须为访问的项目实施自己的缓存逻辑。
请求指标收集	`.withRequestMetricCollector(collector)`	null	在构建标准 DynamoDB 客户端时在 ClientOverrideConfiguration 中使用 metricPublisher() 。
保存行为	`.withSaveBehavior(SaveBehavior.CLOBBER)` 选项包括 UPDATE、CLOBBER、PUT、T 或 UPDATE_SKIP_NULL_ATTRIBUTES 。	UPDATE	在 V2 中，您可以显式调用 putItem() 或 updateItem() 。 CLOBBER or PUT：V2 中的相应动作是调用 putItem() 。没有特定的 CLOBBER 配置。 UPDATE对应于 updateItem() 。 UPDATE_SKIP_NULL_ATTRIBUTES 对应于 updateItem() 。使用请求设置 ignoreNulls 和注释/标签 DynamoDbUpdateBehavior 控制更新行为。 APPEND_SET ：不支持

应用场景	V1	V1 中的默认设置	V2
类型转换器工厂	`.withTypeConverterFactory(typeConverterFactory)`	标准类型转换器	使用以下方式在 bean 上设置 `@DynamoDbBean(converterProviders = {ConverterProvider.class, DefaultAttributeConverterProvider.class})`

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”对象来进行高度灵活的配置。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，不使用配置对象，而是使用生成器对请求对象设置参数。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筛选条件的映射。

应用场景	操作	V1	V2

预期的属性条件	save()、delete()、query()、scan()	`new DynamoDBSaveExpression()`
		`  .withExpected(Collections.singletonMap(`
		`  "otherAttribute", new ExpectedAttributeValue(false)))`
		`  .withConditionalOperator(ConditionalOperator.AND);`
			已弃用；改用 `ConditionExpression` 。

条件表达式	delete()	`deleteExpression.setConditionExpression("zipcode = :zipcode")`
		`deleteExpression.setExpressionAttributeValues(...)`
			`Expression conditionExpression = Expression.builder()`
			`  .expression("#key = :value OR #key1 = :value1")`
			`  .putExpressionName("#key", "attribute")`
			`  .putExpressionName("#key1", "attribute3")`
			`  .putExpressionValue(":value", AttributeValues.stringValue("wrong"))`
			`  .putExpressionValue(":value1", AttributeValues.stringValue("three"))`
			`  .build();`
			
			`DeleteItemEnhancedRequest request = DeleteItemEnhancedRequest.builder()`
			`  .conditionExpression(conditionExpression).build();`

应用场景	操作	V1	V2
筛选表达式	query()、scan()	`scanExpression .withFilterExpression("#statename = :state") .withExpressionAttributeValues(attributeValueMapBuilder.build()) .withExpressionAttributeNames(attributeNameMapBuilder.build())`	`Map<String, AttributeValue> values = singletonMap(":key", stringValue("value")); Expression filterExpression = Expression.builder() .expression("name = :key") .expressionValues(values) .build(); QueryEnhancedRequest request = QueryEnhancedRequest.builder() .filterExpression(filterExpression).build();`
查询的条件表达式	query()	`queryExpression.withKeyConditionExpression()`	`QueryConditional keyEqual = QueryConditional.keyEqualTo(b -> b .partitionValue("movie01")); QueryEnhancedRequest tableQuery = QueryEnhancedRequest.builder() .queryConditional(keyEqual) .build();`

类型转换

默认转换器

在 V2 中，SDK 为所有常见类型提供了一组默认转换器。既可以在整体提供程序级别更改类型转换器，也可以为单个属性更改类型转换器。您可以在 AttributeConverter API 参考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 @DynamoDBTypeConverted 对 getter 方法添加注释，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，CurrencyFormatConverter 可以在 Java Currency 类型和 DynamoDB 字符串之间应用转换，如以下代码段所示。

```
@DynamoDBTypeConverted(converter = CurrencyFormatConverter.class)
public Currency getCurrency() { return currency; }
```

上一个代码段的 V2 等效代码如下所示。

```
@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程序

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在意的类型。类型转换器工厂扩展 DynamoDBTypeConverterFactory，通过获取对默认集的引用并对其进行扩展来完成覆盖。以下代码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 注释提供了类似的功能。您可以提供单个 AttributeConverterProvider 或一个有序的 AttributeConverterProvider 链。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，且必须在链中包括它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。

适用于 Java 的 SDK 版本 1 和版本 2 之间的字符串处理差异

向 DynamoDB 发送数据时，V1 和 V2 对空字符串的处理方式有所不同：

- V1：在发送到 DynamoDB 之前将空字符串转换为 null 值（导致没有属性）
- V2：将空字符串作为实际的空字符串值发送给 DynamoDB

> **Important**
>
> 迁移到 V2 后，如果您不想在 DynamoDB 中存储空字符串，则必须实施自定义转换器。如果没有自定义转换器，V2 会将空字符串作为实际的空字符串属性存储在 DynamoDB 项目中，这与 V1 完全省略这些属性的行为不同。

Example V2 的自定义转换器，用于将空字符串属性转换为 null

```
/**
 * Custom converter that maintains V1 behavior by converting empty strings to null
 values
 * when writing to DynamoDB, ensuring compatibility with existing data. No attribute
 will be saved to DynamoDB.
```

```
 */
public class NullifyEmptyStringConverter implements AttributeConverter<String> {
    @Override
    public AttributeValue transformFrom(String value) {
        if (value == null || value.isEmpty()) {
            return AttributeValue.builder().nul(true).build();
        }
        return AttributeValue.builder().s(value).build();
    }

    @Override
    public String transformTo(AttributeValue attributeValue) {
        if (attributeValue.nul()) {
            return null;
        }
        return attributeValue.s();
    }

    @Override
    public EnhancedType<String> type() {
        return EnhancedType.of(String.class);
    }

    @Override
    public AttributeValueType attributeValueType() {
        return AttributeValueType.S;
    }
}

// V2 usage:
@DynamoDbBean
public class Customer {
    private String name;

    @DynamoDbConvertedBy(NullifyEmptyStringConverter.class)
    public String getName() {
        return name;
    }
}
```

适用于 Java 的 SDK 版本 1 和版本 2 之间的乐观锁差异

V1 和 V2 都使用属性注释来实施乐观锁，注释在 bean 类上标记一个属性以存储版本号。

乐观锁行为的差异

	V1	V2
bean 类注释	`@DynamoDBVersionAttribute`	`@DynamoDbVersionAttribute`（请注意，V2 使用小写的“b”）
初始保存	版本号属性设置为 1。	使用 `@DynamoDbVersionAttribute(startAt = X)` 设置的版本属性的起始值。默认值为 0。
更新	如果条件检查确认正在更新的对象的版本号与数据库中的版本号相匹配，则版本号属性将增加 1。	如果条件检查确认正在更新的对象的版本号与数据库中的版本号相匹配，则版本号属性会递增。 按照使用 `@DynamoDbVersionAttribute(incrementBy = X)` 设置的 `incrementBy` 选项来增加版本号属性。默认值是 1。

删除	`DynamoDBMapper` 添加条件检查，以确认要删除的对象的版本号与数据库中的版本号相匹配。	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。
		
		在以下示例中，`recordVersion` 是 bean 的版本属性。
		
		`// 1. Read the item and get its current version.`
		`Customer item = customerTable.getItem(Key.builder().partitionValue("someId").build());`
		`AttributeValue currentVersion = item.getRecordVersion();`
		
		`// 2. Create conditional delete with the `currentVersion` value.`
		`DeleteItemEnhancedRequest deleteItemRequest =`
		`DeleteItemEnhancedRequest.builder()`
		`.key(KEY)`
		`.conditionExpression(Expression.builder()`
		`.expression("recordVersion = :current_version_value")`
		`.putExpressionValue(":current_version_value", currentVersion)`
		`.build()).build();`

	V1	V2
		`customerTable.deleteItem(deleteItemRequest);`
带条件检查的事务写入	您不能使用在 `addConditionCheck` 方法中用 `@DynamoDBVersionAttribute` 进行注释的 bean 类。	您可以在 `addConditionCheck` 生成器方法中通过 `@DynamoDbVersionAttribute` 注释为 `transactWriteItems` 请求使用 bean 类。
禁用	通过将 `DynamoDBMapperConfig.SaveBehavior` 枚举值从 `UPDATE` 更改为 `CLOBBER` 来禁用乐观锁。	不使用 `@DynamoDbVersionAttribute` 注释。

适用于 Java 的 SDK 版本 1 和版本 2 之间的 fluent setter 方法差异

您可以在 DynamoDB 映射 API 的 V1 版以及自 2.30.29 版本开始的 V2 版中使用带 fluent setter 的 POJO。

例如，以下 POJO 通过 `setName` 方法返回一个 `Customer` 实例：

```
// V1

@DynamoDBTable(tableName ="Customer")
public class Customer{
  private String name;
  // Other attributes and methods not shown.
  public Customer setName(String name){
```

```
      this.name = name;
      return this;
  }
}
```

但是，如果您使用的是 2.30.29 之前的 V2 版本，则 `setName` 会返回 `Customer` 实例，其 `name` 值为 `null`。

```
// V2 prior to version 2.30.29.

@DynamoDbBean
public class Customer{
  private String name;
  // Other attributes and methods not shown.
  public Customer setName(String name){
      this.name = name;
      return this;  // Bug: returns this instance with a `name` value of `null`.
  }
}
```

```
// Available in V2 since version 2.30.29.

@DynamoDbBean
public class Customer{
  private String name;
  // Other attributes and methods not shown.
  public Customer setName(String name){
      this.name = name;
      return this;  // Returns this instance for method chaining with the `name` value
 set.
  }
}
```

AWS SDK for Java 版本 1 和版本 2 之间的文档 API 差异

文档 API 支持将 JSON 风格的文档作为 DynamoDB 表中的单个项目来处理。V1 文档 API 在 V2 中有对应的 API，但是 V2 没有像 V1 那样为文档 API 使用单独的客户端，而是在 DynamoDB 增强型客户端中整合了文档 API 功能。

在 V1 中，`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 `EnhancedDocument` 类的实例表示。请注意，在 V2 中，主键在表架构中定义，在 V1 中，主键在项目本身定义。

下表比较了 V1 和 V2 中文档 API 之间的差异。

应用场景	V1	V2
Create a document client	`AmazonDynamoDB client = ... //Create a client. DynamoDB documentClient = new DynamoDB(client);`	`// The V2 Document API uses the same DynamoDbEnhancedClient // that is used for mapping POJOs. DynamoDbClient standardClient = ... //Create a standard client. DynamoDbEnhancedClient enhancedClient = ... // Create an enhanced client.`
Reference a table	`Table documentTable = docClient.documentClient("Person");`	`DynamoDbTable<EnhancedDocument> documentTable = enhancedClient.table("Person", TableSchema.documentSchemaBuilder() .addIndexPartitionKey(TableMetadata.primaryIndexName(),"id", AttributeValueType.S) .attributeConverterProviders(AttributeConverterProvider.defaultProvider())`

应用场景	V1	V2
		.build()) ;
Work with semi-structured data		
Put item	Item item = new Item() .withPrimaryKey("id", 50) .withString("firstName", "Shirley"); PutItemOutcome outcome = documentTable.putItem(item);	EnhancedDocument personDocument = EnhancedDocument.builder() .putNumber("id", 50) .putString("firstName", "Shirley") .build(); documentTable.putItem(personDocument);
Get item	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		

应用场景	V1	V2
Convert a JSON structure to use it with the Document API	`// The 'jsonPerson' identifier is a JSON string. Item item = new Item().fromJSON(jsonPerson);`	`// The 'jsonPerson' identifier is a JSON string. EnhancedDocument document = EnhancedDocument.builder() .json(jsonPerson).build());`
Put JSON	`documentTable.putItem(item)`	`documentTable.putItem(document);`
Read JSON	`GetItemOutcome outcome = //Get item. String jsonPerson = outcome.getItem().toJSON();`	`String jsonPerson = documentTable.getItem(Key.builder() .partitionValue(50).build()) .fromJson();`

API 参考和文档 API 指南

	V1	V2
API 参考	API 参考	API 参考
文档指南	Amazon DynamoDB 开发人员指南	增强型文档 API（本指南）

V1 Xpec API 到 V2 表达式 API

V1 中提供的表达式规范（Xspec）API 协助创建用于处理面向文档的数据的表达式，该 API 在 V2 中不可用。V2 使用表达式 API，该 API 可以处理面向文档的数据和对象到项目的映射数据。

	V1	V2
API 名称	表达式规范（Xspec）API	表达式 API
结合使用	文档 API 表类的方法，例如 updateItem 和 scan	DynamoDB 增强型客户端的两个 API： 1. 对象到项目映射 API 的方法 2. 用于处理面向文档的数据（JSON）的增强型文档 API 的方法 对于这两种类型的 API，在您购买 DynamoDbTable 实例后： • 例如，来自对象到项目的映射 API 的 BeanTableSchema • 来自面向文档的 API 的 DocumentTableSchema 创建请求对象时，在 DynamoDbTable 方法中使用表达式。例如，在 QueryEnhancedRequest.Builder 的 filterExpression 方法中
资源	• Xpec 首次发布的博客文章 • API 参考	本 Java 开发人员指南中的表达式信息

加密库迁移

有关迁移 DynamoDB 加密库以与 Java SDK 的 V2 配合使用的信息，请参阅《Amazon DynamoDB Encryption Client 开发人员指南》。

Amazon SQS 请求批处理从版本 1 到版本 2 的更改

本主题详细介绍了AWS SDK for Java 版本 1 和版本 2 之间的 Amazon SQS 自动请求批处理的更改。

高级别更改

AWS SDK for Java 1.x 使用单独的 AmazonSQSBufferedAsyncClient 类执行客户端缓冲，该类需要显式初始化才能进行请求批处理。

AWS SDK for Java 2.x 使用 SqsAsyncBatchManager 简化并增强了缓冲功能。该接口的实现提供了直接与标准 SqsAsyncClient 集成的自动请求批处理功能。要了解 v2 的 SqsAsyncBatchManager，请参阅本指南中的 the section called “使用自动请求批处理” 主题。

更改	v1	v2
Maven 依赖项	<dependencyManagement> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-bom</ artifactId> <version> *1.12.782*[1]</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> com.amazonaws</gro upId> <artifact Id>aws-java-sdk-sqs</ artifactId> </dependency> </dependencies>	<dependencyManagement> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>bom</artifactId> <version> *2.31.15*[2]</version> <type>pom</ type> <scope>im port</scope> </dependency> </dependencies> </dependencyManageme nt> <dependencies> <dependency> <groupId> software.amazon.aw ssdk</groupId> <artifact Id>sqs</artifactId> </dependency> </dependencies>

更改	v1	v2
软件包名称	com.amazonaws.services.sqs.buffered	software.amazon.awssdk.services.sqs.batchmanager
类名	AmazonSQSBufferedAsyncClient	SqsAsyncBatchManager

[1] 最新版本。[2] 最新版本。

使用 SQS 自动请求批处理

更改	v1	v2
创建批量管理器	`AmazonSQSAsync sqsAsync = new AmazonSQSAsyncClient(); AmazonSQSAsync bufferedSqs = new AmazonSQSBufferedAsyncClient(sqsAsync);`	`SqsAsyncClient asyncClient = SqsAsyncClient.create(); SqsAsyncBatchManager sqsAsyncBatchManager = asyncClient.batchManager();`
使用自定义配置创建批量管理器	`AmazonSQSAsync sqsAsync = new AmazonSQSAsyncClient();  QueueBufferConfig queueBufferConfig = new QueueBufferConfig() .withMaxBatchOpenMs(200) .withMaxBatchSize(10)`	`BatchOverrideConfiguration batchOverrideConfiguration = BatchOverrideConfiguration.builder() .sendRequestFrequency(Duration.ofMillis(200)) .maxBatchSize(10) .receiveMessageMinWaitDuration(Duration.ofMillis(1000))`

更改	v1	v2
	.withMinReceiveWaitTimeMs(1000) .withVisibilityTimeoutSeconds(20) .withReceiveMessageAttributeNames(messageAttributeValues); AmazonSQSAsync bufferedSqs = new AmazonSQSBufferedAsyncClient(sqsAsync, queueBufferConfig);	.receiveMessageVisibilityTimeout(Duration.ofSeconds(20)) .receiveMessageSystemAttributeNames(messageSystemAttributeNames) .receiveMessageAttributeNames(messageAttributeValues) .build(); SqsAsyncBatchManager sqsAsyncBatchManager = SqsAsyncBatchManager.builder() .overrideConfiguration(batchOverrideConfiguration) .client(SqsAsyncClient.create()) .scheduledExecutor(Executors.newScheduledThreadPool(8)) .build();

更改	v1	v2
发送消息	`Future<SendMessageResult> sendResultFuture = bufferedSqs.sendMessageAsync(new SendMessageRequest() .withQueueUrl(queueUrl) .withMessageBody(body));`	`CompletableFuture<SendMessageResponse> sendCompletableFuture = sqsAsyncBatchManager.sendMessage( SendMessageRequest.builder() .queueUrl(queueUrl) .messageBody(body) .build());`
删除消息	`Future<DeleteMessageResult> deletResultFuture = bufferedSqs.deleteMessageAsync(new DeleteMessageRequest() .withQueueUrl(queueUrl));`	`CompletableFuture<DeleteMessageResponse> deleteResultCompletableFuture = sqsAsyncBatchManager.deleteMessage( DeleteMessageRequest.builder() .queueUrl(queueUrl) .build());`

更改	v1	v2
更改消息的可见性	`Future<ChangeMessageVisibilityResult> changeVisibilityResultFuture = bufferedSqs.changeMessageVisibilityAsync(new ChangeMessageVisibilityRequest() .withQueueUrl(queueUrl) .withVisibilityTimeout(20));`	`CompletableFuture<ChangeMessageVisibilityResponse> changeResponseCompletableFuture = sqsAsyncBatchManager.changeMessageVisibility( ChangeMessageVisibilityRequest.builder() .queueUrl(queueUrl) .visibilityTimeout(20) .build());`
接收消息	`ReceiveMessageResult receiveResult = bufferedSqs.receiveMessage(new ReceiveMessageRequest() .withQueueUrl(queueUrl));`	`CompletableFuture<ReceiveMessageResponse> responseCompletableFuture = sqsAsyncBatchManager.receiveMessage( ReceiveMessageRequest.builder() .queueUrl(queueUrl) .build());`

异步返回类型差异

更改	v1	v2
返回类型	`Future<ResultType>`	`CompletableFuture<ResponseType>`
回调机制	使用单独的 onSuccess 和 onError 方法需要 AsyncHandler	使用 JDK 提供的 CompletableFuture API，例如 whenComplete() 、thenCompose() 、thenApply()
异常处理	使用 AsyncHandler#onError() 方法	使用 JDK 提供的 CompletableFuture API，例如 exceptionally() 、handle() 或 whenComplete()
取消	通过 Future.cancel() 实现基本支持	取消父 CompletableFuture 会自动取消链中的所有相关 Future

异步完成处理差异

更改	v1	v2
响应处理程序实现	`Future<ReceiveMessageResult> future = bufferedSqs.receiveMessageAsync( receiveRequest, new AsyncHandler<ReceiveMessageRequest, ReceiveMessageResult>() { @Override`	`CompletableFuture<ReceiveMessageResponse> completableFuture = sqsAsyncBatchManager .receiveMessage(ReceiveMessageRequest.builder() .queueUrl(queueUrl).build())`

更改	v1	v2
	public void onSuccess(ReceiveMessageRequest request, ReceiveMessageResult result) { List<Message> messages = result.getMessages(); System.out.println("Received " + messages.size() + " messages"); for (Message message : messages) { System.out.println("Message ID: " + message.getMessageId()); System.out.println("Body: " + message.getBody()); } } @Override public void onError(Exception e) { System.err.println("Error receiving messages: " + e.getMessage()); e.printStackTrace(); }	.whenComplete((receiveMessageResponse, throwable) -> { if (throwable != null) { System.err.println("Error receiving messages: " + throwable.getMessage()); throwable.printStackTrace(); } else { List<Message> messages = receiveMessageResponse.messages(); System.out.println("Received " + messages.size() + " messages"); for (Message message : messages) { System.out.println("Message ID: " + message.messageId()); System.out.println("Body: " + message.body()); } } });

更改	v1	v2
	});	

关键配置参数

参数	v1	v2
最大批次大小	maxBatchSize （每批默认为 10 个请求）	maxBatchSize （每批默认为 10 个请求）
批次等待时间	maxBatchOpenMs （默认值为 200 毫秒）	sendRequestFrequency （默认值为 200 毫秒）
可见性超时	visibilityTimeoutSeconds （队列默认值为 -1）	receiveMessageVisibilityTimeout （队列默认值）
最小等待时间	longPollWaitTimeoutSeconds （如果 longPoll 为 true，则为 20 秒）	receiveMessageMinWaitDuration （默认值为 50 毫秒）
消息属性	使用 ReceiveMessageRequest 进行设置	receiveMessageAttributeNames （默认值为无）
系统属性	使用 ReceiveMessageRequest 进行设置	receiveMessageSystemAttributeNames （默认值为无）
长轮询	longPoll（默认值为 true）	不支持，以便避免让已建立的连接一直处于等待状态，直到服务器发送消息
长时间轮询的最长等待时间	longPollWaitTimeoutSeconds （默认值为 20 秒）	不支持，以便避免让已建立的连接一直处于等待状态，直到服务器发送消息

参数	v1	v2
在客户端预取和存储接收批次的最大数量	maxDoneReceiveBatches （10 个批次）	不支持，因为这在内部处理
同时处理的最大活动出站批次数量	maxInflightOutboundBatches （默认为 5 个批次）	不支持，因为这在内部处理
同时处理的最大活动接收批次数量	maxInflightReceiveBatches （默认为 10 个批次）	不支持，因为这在内部处理

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 2.27.21 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>2.27.21</version>
          <type>pom</type>
          <scope>import</scope>
        </dependency>
```

```
    </dependencies>
</dependencyManagement>

<dependencies>
    <dependency>
      <groupId>com.amazonaws</groupId>
      <artifactId>aws-java-sdk-s3</artifactId>
    </dependency>
    <dependency>
      <groupId>software.amazon.awssdk</groupId>
      <artifactId>dynamodb</artifactId>
    </dependency>
</dependencies>
```

查找使用AWS SDK for Java 1.x 客户端的应用程序

在迁移到AWS SDK for Java 2.x 之前，您需要确定您环境中的哪些应用程序使用版本 1.x 客户端。您可以在 AWS CloudTrail 事件中查询“userAgent”字段来查找这些应用程序。

使用 CloudTrail Lake 查找具有 1.x 客户端的应用程序

借助 AWS CloudTrail Lake，您可以查询 CloudTrail 记录的事件。按照以下步骤来创建数据湖，用于识别应用程序使用的 SDK 版本：

1. 创建 CloudTrail 数据湖。要创建事件数据存储，请参阅用户指南。
2. 创建数据存储后，检查记录内容。记录正文包含用于确定请求的操作、计时和位置的字段。有关详细信息，请参阅 CloudTrail 记录内容的用户指南。
3. 对您的数据运行查询。按照用户指南来查询和保存查询结果。

每条记录中的 userAgent 字段包含发出请求的 SDK 版本。使用此字段来识别使用 Java SDK 版本 1.x 的应用程序。

以下示例查询用于查找从 2025 年 6 月 17 日起，针对 EventDatastoreID `sample-Data-Store-Id` 使用 Java SDK 1.x 发起的所有请求：

```
select userIdentity, eventSource, awsRegion,
    eventName, eventType, eventTime, userAgent,
    requestParameters, sourceIPAddress
 from sample-Data-Store-Id
```

```
where eventTime > '2025-06-17 00:00:00'
and userAgent like '%aws-sdk-java/1.%'
order by eventTime desc
```

查询结果中的事件内容类似于以下示例：

```
{
    "userIdentity": "{
         "type": "IAMUser",
         "principalId": "AIDAJ45Q7YFFAREXAMPLE",
         "arn": "arn:aws:iam::123456789012:user/Alice",
         "accountId": "123456789012",
         "accessKeyId": "",
         "userName": "Alice"
    }",
    "eventSource": "dynamodb.amazonaws.com",
    "awsRegion": "us-west-2",
    "eventName": "ListTables",
    "eventType": "AwsApiCall",
    "eventTime": "2025-07-01 02:23:52.000",
    "userAgent": "aws-sdk-java/1.12.746 Linux/5.10.240 OpenJDK/11.0.25+9-LTS ...",
    "requestParameters": "",
    "sourceIPAddress": "12.345.6.78"
}
```

您可以使用此信息来帮助确定何时何地发出请求。

在示例中，使用名为 Alice 的 IAM 用户的凭证，在 2025-07-01 02:23:52 (UTC) 从 IP 地址 12.345.6.78 发出 DynamoDB ListTables 请求。userAgent 字段的值表明，请求是使用AWS SDK for Java 版本 1.12.746 从装有 JDK 11 的 Linux 系统发出。

有关 AWS CloudTrail 事件记录中各字段的描述，请参阅 CloudTrail 记录管理、数据和网络活动事件的内容。

为用户提供安全保障 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。安全是双方共同承担 AWS 的责任。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
- 在适用于 Java 的 SDK 中使用 TLS
- 身份和访问管理
- 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结构上的内容的控制。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，请参阅数据隐私常见问题。有关欧洲数据保护的信息，请参阅 AWS Security Blog 上的 AWS Shared Responsibility Model and GDPR 博客文章。

出于数据保护目的，我们建议您保护 AWS 账户 凭证并使用 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。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）。

- 用于 SSL/TLS 与 AWS 资源通信。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用设置 API 和用户活动日志 AWS CloudTrail。有关使用 CloudTrail 跟踪捕获 AWS 活动的信息，请参阅AWS CloudTrail 用户指南中的使用跟 CloudTrail 踪。
- 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），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果您在 AWS 通过命令行界面或 API 进行访问时需要经过 FIPS 140-3 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准（FIPS）第 140-3 版》https://aws.amazon.com/compliance/fips/。

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）。这包括使用控制台、API 或 AWS 服务 使用适用于 Java 的 SDK 或其他软件开发工具包时 AWS SDKs。 AWS CLI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。如果您向外部服务器提供 URL，强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

在适用于 Java 的 SDK 中使用 TLS

AWS SDK for Java 使用其底层 Java 平台的 TLS 功能。在本主题中，我们将演示使用 Amazon Corretto 17 使用的 OpenJDK 实现的示例。

要使用 AWS 服务，底层 JDK 必须支持 TLS 1.2 的最低版本，但建议使用 TLS 1.3。

用户应查阅他们与 SDK 结合使用的 Java 平台的文档，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的 TLS 版本。

如何检查 TLS 版本信息

以下代码演示了如何使用 OpenJDK 来打印SSLContext支持 TLS/SSL 哪些版本。

```
System.out.println(Arrays.toString(SSLContext.getDefault().getSupportedSSLParameters().getProto
```

例如，Amazon Corretto 17 (OpenJDK) 会生成以下输出。

```
[TLSv1.3, TLSv1.2, TLSv1.1, TLSv1, SSLv3, SSLv2Hello]
```

要查看 SSL 握手的运行情况以及使用的 TLS 版本，可使用系统属性 javax.net.debug。

例如，运行使用 TLS 的 Java 应用程序。

```
java app.jar -Djavax.net.debug=ssl:handshake
```

应用程序将记录 SSL 握手，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",
...
```

强制使用最低版本的 TLS

适用于 Java 的 SDK 始终首选平台和服务支持的最新 TLS 版本。如果您希望强制指定特定的最低 TLS 版本，请查阅 Java 平台的文档。

对于基于 OpenJDK 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

有关迁移到 TLS 1.2 的最低版本 AWS 的 API 端点的信息，请参阅此博客文章。

身份和访问管理

AWS Identity and Access Management (IAM) AWS 服务 可帮助管理员安全地控制对 AWS 资源的访问权限。IAM 管理员控制谁可以进行身份验证（登录）和授权（有权限）使用 AWS 资源。您可以使用 IAM AWS 服务 ，无需支付额外费用。

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

您的使用方式 AWS Identity and Access Management (IAM) 会有所不同，具体取决于您所做的工作 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 管理员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。要详细了解您的公司如何使用 IAM 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，您可能希望了解如何编写策略以管理对 AWS的访问权限的详细信息。要查看您可以在 IAM 中使用的 AWS 基于身份的策略示例，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 IAM 用户身份进行身份验证 AWS 账户根用户，或者通过担任 IAM 角色进行身份验证。

您可以使用来自身份源的证书 AWS IAM Identity Center（例如（IAM Identity Center）、单点登录身份验证或 Google/Facebook 证书，以联合身份登录。有关登录的更多信息，请参阅《AWS 登录 用户指南》中的如何登录您的 AWS 账户。

对于编程访问，AWS 提供 SDK 和 CLI 来对请求进行加密签名。有关更多信息，请参阅《IAM 用户指南》中的适用于 API 请求的AWS 签名版本 4。

AWS 账户 root 用户

创建时 AWS 账户，首先会有一个名为 AWS 账户 root 用户的登录身份，该身份可以完全访问所有资源 AWS 服务 和资源。我们强烈建议不要使用根用户进行日常任务。有关需要根用户凭证的任务，请参阅《IAM 用户指南》中的需要根用户凭证的任务。

联合身份

作为最佳实践，要求人类用户使用与身份提供商的联合身份验证才能 AWS 服务 使用临时证书进行访问。

联合身份是指来自您的企业目录、Web 身份提供商的用户 Directory Service ，或者 AWS 服务 使用来自身份源的凭据进行访问的用户。联合身份代入可提供临时凭证的角色。

要集中管理访问权限，建议使用。 AWS IAM Identity Center有关更多信息，请参阅《AWS IAM Identity Center 用户指南》中的什么是 IAM Identity Center？。

IAM 用户和群组

IAM 用户是对某个人员或应用程序具有特定权限的一个身份。建议使用临时凭证，而非具有长期凭证的 IAM 用户。有关更多信息，请参阅 IAM 用户指南中的要求人类用户使用身份提供商的联合身份验证才能 AWS 使用临时证书进行访问。

IAM 组指定一组 IAM 用户，便于更轻松地对大量用户进行权限管理。有关更多信息，请参阅《IAM 用户指南》中的 IAM 用户使用案例。

IAM 角色

IAM 角色是具有特定权限的身份，可提供临时凭证。您可以通过从用户切换到 IAM 角色（控制台）或调用 AWS CLI 或 AWS API 操作来代入角色。有关更多信息，请参阅《IAM 用户指南》中的担任角色的方法。

IAM 角色对于联合用户访问、临时 IAM 用户权限、跨账户访问、跨服务访问以及在 Amazon 上运行的应用程序非常有用。 EC2有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

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。策略定义了与身份或资源关联时的权限。 AWS 在委托人提出请求时评估这些政策。大多数策略都以 JSON 文档的 AWS 形式存储在中。有关 JSON 策略文档的更多信息，请参阅《IAM 用户指南》中的 JSON 策略概述。

管理员使用策略，通过定义哪个主体可以在什么条件下对哪些资源执行哪些操作来指定谁有权访问什么。

默认情况下，用户和角色没有权限。IAM 管理员创建 IAM 策略并将其添加到角色中，然后用户可以担任这些角色。IAM 策略定义权限，与执行操作所用的方法无关。

基于身份的策略

基于身份的策略是您附加到身份（用户、组或角色）的 JSON 权限策略文档。这些策略控制身份可以执行什么操作、对哪些资源执行以及在什么条件下执行。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户管理型策略定义自定义 IAM 权限。

基于身份的策略可以是内联策略（直接嵌入到单个身份中）或托管策略（附加到多个身份的独立策略）。要了解如何在托管策略和内联策略之间进行选择，请参阅《IAM 用户指南》中的在托管策略与内联策略之间进行选择。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。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。您必须在基于资源的策略中指定主体。

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中使用 IAM 中的 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 和 Amazon VPC 就是支持的服务示例 ACLs。 AWS WAF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 支持其他策略类型，这些策略类型可以设置更常见的策略类型授予的最大权限：

- 权限边界 – 设置基于身份的策略可以授予 IAM 实体的最大权限。有关更多信息，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)-在中指定组织或组织单位的最大权限 AWS Organizations。有关更多信息，请参阅《AWS Organizations 用户指南》中的服务控制策略。

- 资源控制策略 (RCPs)-设置账户中资源的最大可用权限。有关更多信息，请参阅《AWS Organizations 用户指南》中的资源控制策略 (RCPs)。
- 会话策略 – 在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。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 用户指南中的与 IAM 配合使用的AWS 服务。

要了解如何在 IAM 中 AWS 服务 使用特定的，请参阅相关服务的《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 IAM 时可能遇到的常见问题。

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，指明您无权执行某个操作，则必须更新策略以允许执行该操作。

当 mateojackson IAM 用户尝试使用控制台查看有关虚构 *my-example-widget* 资源的详细信息，但不拥有虚构 awes:*GetWidget* 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 awes: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 awes:*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 marymajor 的 IAM 用户尝试使用控制台在 AWS中执行操作时，会发生以下示例错误。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，可以代入角色。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 AWS 账户 ，请参阅 IAM 用户指南中的向您拥有 AWS 账户 的另一个 IAM 用户提供访问权限。
- 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
- 要了解如何通过身份联合验证提供访问权限，请参阅《IAM 用户指南》中的为经过外部身份验证的用户（身份联合验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

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。有关您在使用时的合规责任的更多信息 AWS 服务，请参阅AWS 安全文档。

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 提供多个物理隔离和隔离的可用区，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、容错能力和可扩展性。

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 ecurity Pillar Well-Architected Fram ework 中的基础设施保护。

您可以使用 AWS 已发布的 API 调用通过网络访问此 AWS 产品或服务。客户端必须支持以下内容：

- 传输层安全性协议（TLS）。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 具有完全向前保密（PFS）的密码套件，例如 DHE（临时 Diffie-Hellman）或 ECDHE（临时椭圆曲线 Diffie-Hellman）。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 主体关联的秘密访问密钥来对请求进行签名。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合 AWS 规工作范围内的AWS 服务。

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	2026-09-27
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-----
Comment: Hostname:
Version: Hockeypuck 2.2

xsFNBFd1gAUBEACqbmmFbxdJgz1lD7wrlskQA1LLuSAC4p8ny9u/D2zLR8Ynk3Yz
mzJuQ+Kfjne2t+xTDex6MPJlMYpOviSWsX2psgvdmeyUpW9ap0lrThNYkc+W5fRc
buFehfbi9LSATZGJi8RG0sCCr5FsYVz0gEk85M2+PeM24cXhQIOZtQUjswX/pdk/
KduGtZASqNAYLKROmRODzUuaokLPo24pfm9bnr1RnRtwt5ktPAA5bM9ZZaGKriej
kT2lPffbBjp8F5AZvmGLtNm2Cmg4FKBvI04SQjy2jjrQ3wBzi5Lc9HTxDuHK/rtV
u6PewUe2WPlnxlXenhMZU1UK4YoSB9E9StQ2VxQiySLHSdxR7Ma4WgYdVLn9bOie
nj3QxLuQ1ZUKF79ES6JaM4tOz1gGcQeU1+UklgjFLuKwmzWRdEIFfxMyvH6qgKnd
U+DioH5mcUwhwffAAsuIJyAdMIEUYh7IfzJJXQf+fF+XfOCl6byOJFWrIGQkAzMu
```

```
CEvaCfwtHC2Lpzo33/WRFeMAuzzd0QJ4uz4xFFvaSOSZHMLHWI9YV/+Pea3X99Ms
0Nlek/LolAJh67MynHeVBOHKrq+fluorWepQivctzN6Y1NOkx5naTPGGaKWK7G2q
TbcY5SMnkIWfLFSougj0Fvmjczq8iZRwYxWA+i+LQvsR9WEXEiQffIWRoQARAQAB
zSxBV1MgU0RLcyBhbmQgVG9vbHMgPGF3cy1kci10b29sc0BhbWF6b24uY29tPsLB
lAQTAQoAPgIbAwULCQgHAwUVCgkICwUWAgMBAAIeAQIXgBYhBP65IJ8vLz9GZIQe
VawQezhmktrdBQJo12ZrBQkTQxnmAAoJEKwQezhmktrdi18P/A3De83MBx8bdcWJ
Fot71Vk1TyBQFErgrtcytSU0czEHx3tGbzgQLbMlyzjirOT03usxEkOeqTVK+RU+
5uFXNZYQLwMJlHJ6S8tnfLe/ExM5WQ2KPwIUPfZs1GDDRQB2dIKSc+qYrP1O1vf4
O4iPgfLHMW2bFh3zjjxcaHCJyqc7Cau33eZFBAsRni1jOUo7MeyX0hlXfW8pd48Q
wZllQVZ/6KmDiFWA0CZ+2svJ5cL0tgPoh1OQjoz0nHpNfuDILMrZ+e7tx2VTlkGH
UGeNSydnrK8v9ztFn34KtU/k7NEWoVSyEi+5ICZL18FBwPqTwdVWXwXrqZCKiIpr
8ZdJWDz2sJfgDFNCC6rKgCQ6FrmaD9G76dYWkQ4AbZqABlUzU3q36W1K0r3iOAb5
G4tdOt4yqXHTe1x+ZUNaeW7gaCmtXAxLw0OfeJrcq/44b/SQP+qJ8sSOv76Yg2oF
BsF5DWOVUFghbTyokHAoVROyhBR4dUUisY39AqLSL8+Lp9Pr3wNuGl9GLrMD5701
piUb88B3Gwe1EiKV1gaKrvZ3mECDUiSMVO0Z5iG8E4QDpNmVbJbV1uT821ubvtOv
2Ko10Fa0uwCYGssdRGqEXNy6jz/Er8LAC3+nmGINDJQzrF+loYoSSkI2Nu7lhMuL
7iWwUPF7OhDXoVSAn4X3x6q2rGK0wsGUBBMBCgA+AhsDBQsJCAcDBRUKCQgLBRYC
AwEAAh4BAheAFiEE/rkgny8vP0ZkhB5VrBB7OGaS2t0FAmjXZTsFCRNDGLYACgkQ
rBB7OGaS2t0/Ow//YIv51vHtD+kwMmIvk3zpizDHY0zW2dOezAo+C/DsSyC7wDll
Dixw34EQ1yLXH5xLR8CH1zupl3JmmEp1ucdQggoefbidxDl8Fld7tJOD1y3GGnTD
0jAl2ZC+W65Oh+wS1mD1FlaKjMGGkvJf0dA7RtU2T8dv3vt8dsxg76FMFS3+fqlC
FNOAsNTn9zWR1SqBIfkMJK83aq6s/rcEV9VrAYgDgqex58fygB5EuTf842/IF7WZ
Q9gd6fupB0mMZP5YWd2uj/vsBTYakG+mgQwDxZuKPeEzAqnqqS7biSQOUO6Wozlq
Yy4fSczE9GkBAvg0pGmbko+zHvpnjvX/h1CUpC6odvFyOAhZp6zyhs0QWz9thfqV
lU8WlbgJ2atFDn5GUSxF/fe0Yzovlbbs6sbYXuvMG9RiEOuJ1mBbZR3aIdZ1U6Do
BHc/vjc5mWcV7JQSP7i4W/8W7X3UAuN9LdxB+IvF3Cwrgtlw2BWvA5Alco5Tnz8t
P/CIVmBjk+sLme8W4kfLK3IWEbwClOdNnErI/MHRm65A2Y5EMIhwjrOi07SU1Pxa
nPpg3OYJCdvjzdB8QE3/DBiMfOl4dISfKDVEWnfK8mZaYd/BeRm2gUAa9UrqSFCG
BlA7Lg+eLI3US0FvwWJ4j5bBJqgLu+y7crIkiUOPAQuLk3lO+5uYU/I3DuLCwZQE
EwEKAD4CGwMFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AWIQT+uSCfLy8/RmSEHlWs
EHs4ZpLa3QUCZwAXCAUJEWvKgwAKCRCsEHs4ZpLa3ZdTEACMBLg2q9zk8ZH02nDz
Sg5zc8Wlqq8WdxU0Pj8qx4UOrrMca7wyiUvrgoxPW5lh1RVNUeMkDRfu9pSXcOVI
V9LvmYE/WnwKROubgGbsC4T7M/LqV0/AulXil4d7IXcO6l4toa8LTNWtD5bODgrN
gvay1AzCU8kq1Qw1cKZ2gAfvA3Ba7PWyLeUN4HTlGrXcw73G+0CofY1L8wqWxHCJ
29XqQzeTEc6MDEeIlNlVdUcy8Qr5uwkEsl34H9AxS5F1opJ4TqvXiDZsrSRRv57R
XYmRZDWeYT+9PZaMsHXza5qgej7BfATxhYfICsNaY6MK3x6b+nDSKkoZgO+i09zh
1YjpahhQe6G336v/3mRj0dKGCRQ6znQ9ghUaB5z9zfvgH5AOEkTe3l8MqM+j5A6P
VjSBBJAHKejxr7+wKJKIA6P+DqpsYAunzftwUzrLVqb+BZQ+DcTmVrE7OPcMYJD5
QglX/Le+WmWZHI154NXgpWWUOUgZUbUge4DKrT+zCJ9iecPLKTW7OcULyXO+rjb8
8BGrD5GPlHB3dOUXXTlMKCqg3qy1Bu2KnZTQiaEEdZgSIGQbrW0JTMmmXJkKjokd
JMA4vYeg5en51G9nRQjScPngx77IxvByNyFWTJdG1ENpJpsK9TtmENcpyUJtJZTJ
ZSOIRVPP5RzR5vInuXWq6VV0BMLBlAQTAQoAPgIbAwULCQgHAwUVCgkICwUWAgMB
AAIeAQIXgBYhBP65IJ8vLz9GZIQeVawQezhmktrdBQJlJEoiBQkPj/2dAAoJEKwQ
ezhmktrdx1YP/0vvym3jgX/pwnR7K1rafZMb1iKQBr0ISG8cdbaf4pqX5vuUZnyj
```

```
w9Cl/oONn7jJjnQxOIIzuBoxne2WN28ftM2w0nVXm85mAmz2fwQz/fdKDyonXcOh
pfD2iMqn7gESjhEgRE7wMDYMDuLdqHI7OKWGVfgrh7xEmKapLh45h7cnumo2VjL9
uDYY1aOBHz993T7oE41y43rhk+6kKbGFd2uuo7h5j1ZF8Lj6sYfcEzXOU1OhRlD0
nyBjDy9MYWu0YNouc70WgMceGx6hjvCAM/5fxP7SZFecZ7ePeB0GpvVA24hSNENE
0r3tUekuOf1I0FunMnMnbh7ZO9rPYqWvWDNIpU3S4CjFhY82L+IeKnmLy8N6ASRk
HsPiNCOHSK8C/0ynrd9xLhX8Jsk/TGiQYaleoHhWkNLlZsL86QHL8SKEqkqZCQf5
AEqghDP6NEGS7lnOenA7JjIrA9KLlT7fnNWZOwFi5X+o/CymE2ytEMS0Yf3nmY4U
n9x56Wgn6J2zqB5nqOXf6NxGdAIgOBm098YEnKCIFzk+yhoDlprVpHcnd2b5f6Oq
uh8KYOEbKgpMJ3zZuWSL5kwGF1nNoYiAkonMaz9H3pOQnOMVYCUeUTDRsiO/prrd
UhNlry4TAsBMpeXnFhdLVM3vFQZVpByadGOJNmnaN/Wavw2a00UGBFa4wsF9BBMB
CgAnAhsDBQsJCAcDBRUKCQgLBRYCAwEAAh4BAheABQJhMqGaBQkLnlUVAAoJEKwQ
ezhmktrd2sQP/3YHM+U+BbOy1nSEAfykZ71+uCM2hkHMLdxQYWB/rBWkmg/pbu+d
r4t45RsTASrNjRcZOntlPMQRIq973ymHfpmeS+noFwvTGH7zDv1BRBR9wPrd1XUz
iSuEUHGi/fqxUVXQ5mbonzfThX8tuXeuiQmeToqoB00FYlZm6xsNnEHcjVl66mC4
IPoJLWnZJs4rOCeoRf5XvDTgX6xt5/kLYRZf79qaWGFvaZpsc1CH+rQJUdVa/D4T
7pI7hX6zy0S91z4iuC5HZUiOTF+y5auEZHGTdTWNS1kvOvfcCTi0XK/GkGL82SZu
7X2VGnpCeUnFyViRGlk+KaDG1sVyDY+lcBPg6ilr45M6MQV0iHS5OF04QNXSKt5+
UnzJH7lldgNsR6ibRMyNV3k5v3fyUcSBvIYyLORTTBiVEjQDSbk1QNqbrQlX9CWz
+EJWn16BFTmMFvxBSWPm640GncHP5J3/0MbMw3Cm90x7k8UfNANIemcrJrSxIDwm
g9cVAg3a+D+wxjrVe8jGg0ejvECpm+0yswigj5x6Lqj09A4UgdjEauN+/pn0nhBo
Gv7DzMXtM/LoDtgp6wn93qZVN2TsuHnkEk4UyntB6eWJbBdXHWUr47exiWh0dvQN
tpwCWPT6I7ZTPtA5K/zx+q9m6797BLgAkTYc6gloQL3vs1Z1S3m/hZNawsF9BBMB
CgAnAhsDBQsJCAcDBRUKCQgLBRYCAwEAAh4BAheABQJgmrz2BQkLBnBxAAoJEKwQ
ezhmktrd36oP/2rB2EkwSOCKC4m0heWSfDWi6OBKoEbbDtFtc6/HwqBW8SPsiKlq
zV0e3qBY/LVju04+ktJEK+EGXLnC3iC36MegrQ8zt391kEx/Zv9LIuVOCX90QIAX
dL8MVUkkjRLCFFH8pTgRy1cJYWk1X4dYdXWYc29fCwNVartNdNBhsb2ht3VJeKDE
kUivBHmkjuISDPEnI1coY7Lj0ZtY5cHdRF2eZpB0RkTBpsIt18rCYyHkERZrhmvb
j3rOyPyvOa+1/dQS8/hv5pEmbKx8cy8RdJkmbUHYatPBsjHkJSWr7O7G9VFW4GoN
9CRAI4KkbDSEDjCL5dv2pq0Sew1MkLuWJGULAMgiIUlWcOs5SZZGFSksNQrtSFV9
Z/wGocecMGkGQNXQ06JV/Fry/TvyphBlmylEqL+NLqEcEjnlz90IVu+ZA+M09J96
UlHO7V5GvBgM+QK/q/dJeMHPWrNlo1gA6Nwl/HBdM0DqzdZ2jEPvsQSABvZrPMty
+BAqEar4wqY1AH4X5ccEjO7nJQoBQSDRSki1fkBsc1nx44N/m0kHdIa0Z/Y+Mw4v
WiZhREkOospG1I4lBa3CNTVAhSs9msGsYfkqvFJGHL7sZY8XSv82GBBvA0nUNrsJ
bLBwo2FaQG9eoatRAGkqp4b/OtNtBuGeiQoNwFGbfUZTAaStj5/zZj0swsF9BBMB
CgAnAhsDBQsJCAcDBRUKCQgLBRYCAwEAAh4BAheABQJe+9bwBQkJZ4plAAoJEKwQ
ezhmktrd+ScP/RoaUKriVVAgLHOGs+/mnfKtnfTlClzi5dsdI9/6H0vLpmSWK/Cl
2cT6gary45VMgAeVK+H1lQXafYj+FY++I5kYoe2GrSvIXhpjaFAJyNf/dKleTsqR
Tm371i8b3FDYs5kvy2CnTbmHB8MsOGxck8/YHd1x+g8WpO2IgF89yYCSF3CAdxC3
6bHbs6Z3C3lcM/3SoWF+Yie2P8XeBMPCGp/BcjQzUcHF6G06TwDDYhixucUi6vEY
EH5JtOwVVQ7bubT8OFeOoJwVxlzYz4UoqxjKDWymarTzu03AUIT0PXPece94bJAK
mSh68ItQe3H8tSPMubERWz2tEV3lVkChDGXcC7BYQmxHseolxz/qzCtJ0iX9BvZR
dniZNeNJ/Cu8M2pDp47zdNFXzf/Q/sQ9pQlws22G2g119rWDneBku9n1vTP80/er
SB+VLTBjDiArlCY5y9+BG8wbscExJySoQxkB9j/nlMzPY5rgk0SyxsNj9GbqH+hr
EjS3/uacNwSLxGcOT2E9Teot5pfTEO6fQVq+35QhfAlP8c8jze01W/+u+wXu1Ui9
```

```
azRSzYtCHanGyyet6U1mlBpAkqkZzH6t3CA5czc9i6FbzjvFVZnbRUZIRzfISYew
lF5WqgTn2iYVdxagPRvLF5kjd696brGW9d5HwirCVGaK04VsXWlAblB9wsF9BBMB
CgAnBQJXdYAFAhsDBQkHhh+ABQsJCAcDBRUKCQgLBRYCAwEAAh4BAheAAAoJEKwQ
ezhmktrdWigP/3QWl7aO81BUWyby4HEhN4SdAoWGY/FLqO4mCtuplcnMgRUCSiL9
l2BSCTMCtUcdSWtYwOgSChN2mMsdi1U2FNR5HvNunYR/pFdqjfQurf1ZmKVeG5/4
uuKaOxMw9e8pK5uYAfs+O7gr8gu/f6/Drp7NZk3/yVKpf4WCY9oX9TA1q9O/11nN
cwS45U/d7YP+N1YM9cBXa1DnDcdfm0BlykzouAF0qd1Lwi/tmLENvybD3+2c2WsE
rlFZGSa5ZafO0tTIWXh5k6wh5FdRRycrnSyRK3B9N9+yaXfMQ0XpOypa8dqQEnCi
IsngDCJPxtTrhMWKhBFRUMzK/WZTDboTQSQDK+YVRrE4K8MtoZSKwZLV2r9O3TpX
kpbKsPVYmexerfdMeZfjZMF1bC7BmEs7jciH6JjbqAoAPnHzN0481aeNarINSViX
PQWr2mp9qShei2/RavLtx2ZNrvmGW72ZKpF8E3WWUDpBJqFVeGNRvOm3aZj8o/Hl
ewtNjcT4ouJfqlfKiULv+g7ANEMDLQTFDTg5twRdvmZlB7oTBsavf+LwxPIXhH32
IR7TX7VeicMMxmZnmZK2ANT/QBi3laf+ojVHvB+f6D74eLNq0Zqjfi/3UFNYsYjg
E+YgCqEUBpHbl61nOHwGOSsQwfap2uKKlzukD/KxH5SPBC3DYGBI+KCbzsFNBFd1
gAUBEAC8zNArPWb3dPMThL2xAY+fS60vXdB1SkOtYJpDWpFgvo0d+VQ+hV6XulGA
HAS6xG1WHysPT9KejIRSgLG+e9CaM5yhsxNa1WFGUM4Q9ESo3t+a75Go7xHIxgFj
C046/O6Vh3g9N/PREeuG8zkZ3H2v5fmD+ejyPgk4W9sFL00zjRiZD0FKVYR/j9ue
nEC/2NBcLuFy3q6cDfmCoDEOO62kXMnaGz3knzEK/X1SkcjsxRDq7zaQlQ1Kou+3
dICwy4x5SJQ8jl+eeeEvF2C2/dXmDohb57tqUwioohMUQkmCtvZgEHjypUwgp0MT
o25gWxkvJlSJKUOb6b1786WNySIzF2gxqlkkEmBl4RAssQkeXjrSmGwsMDyHNqyJ
eYFusl8sPaSpo+V2n0z+2B070Uq+wmf1S5A5FpegH0PZzzoNZo8I6QxaZje9YSZU
ijGmZIdEBleRVt3Svhi8MYlnasd4bW2RK1sr7plkBf8QRe6biiQRF3KDOSn5CbmX
pAcHJ1ZHzRRdkXZDNQC6vCJxsy13O0TrhJtAV1Yq347uyUbVi291ISVgroUVtprs
mHoEk5GoOTHbg9SCSt+xi/FiJQC+ubWmIGXoFKMR3UmhDnnzobKcbnbs/Hd981Fd
VghYYvq//gTAkJk0WxfGqO30wtXRndPOA0T+qhP3TE+LtGRJ+wARAQABwsF8BBgB
CgAmAhsMFiEE/rkgny8vP0ZkhB5VrBB7OGaS2t0FAmjXZm4FCRNDGegACgkQrBB7
OGaS2t3y5g/7BFXp/fdanzuQPToJTPen7AVwhLloKaiYhG3GjdXfMPLvu6UtaaGm
qynLolUNNooobptFqc1G9BKoAghQrta7CsDHtsQF2xyc3Mfu0gmpL/7X5a7sFIeJ
j08UjfweHx4DSG4LEZgNaAoWFjZltp4+8cqijkAHXt+r+1ayQG4VVHOWyXXqmSH4
9HqtbPcPyRzxdoVLeshZC9jmhHhhKqw/LwGyipWSOUKQDjWarBwdyhNmWCaLvxH1
ndMp4tq8DPGC3G4T9tYAbANrn7nKfZgHebMSzMw9kSp0L6QvwwTDjJyIWz85WyeH
WHeBysDaBOit3XDlehUew27y7N6a9hQSYjnXuwvre5mjDIOqJon/31R6ui2Z1y9P
a+bC11hbLXXh9tLCXRuoOt6thh9Cq5X1a76PPpEv30o3bpsb6l2hbrut1OKezwvK
l7txito/jfMiWfsZHA9O4SoM+8GnmVingHtZ805n1T4RddJvT/vaqplfI6zf7jmf
a69lALP420riFOQcwntNUM5tVmFUZsnFp2YRd4Ls7MiXVjtABahlSbb94l5WSVc0
jrOLDf94edvzk4R8i2Ob8CfVZNqEsTR6bHz8dT7Q+xQzEdjUujyyZY1UUl157Qeb
OsHjhCtuZYCI04X9hZ37nKnZXSxRlRDCnt5BEiyFu2WD1RscUe6PcVDCwXwEGAEK
ACYCGwwWIQT+uSCfLy8/RmSEHlWsEHs4ZpLa3QUCaNdlPQUJE0MYuAAKCRCsEHs4
ZpLa3XCpD/42DrcveE+q2ulrAIYPDlUlHiwIMejqBDRm6zmr1KSAeb4E6/MFcP4s
rXSSscMlrqG6NVynjNCXjD2YzWii68EwoXLJkgoD3r2ifzkV62EX2MIEeNZAVwuy
KNxorzmy6bhuWltRYNK/hITs2AG5orOk9ADEJ8PixKymrWlhesPaWX6Yhp9/tWaC
RHOSRiLbRVaJ+7sqT88urLmkV9Hqx949Zxv4+cgBVUGL6WXKsfWhHjbDMNJnozWB
SZaIJznLApOM8z+1DNrqUYyfR8SkF4IOVmg6HDzoyuseJJ8JvMAlkvT6F9VBq/iE
yeDYdEEQxwHwozKrEx5Ybxl5mntbqwCXy6kHSx2+/3RZWpZQ8K29YP9QEk0KeGF8
```

```
9Vap3jjNrX4u3cuRNQpeblQc4uFn3Nzaj+cVV4YzcRw94NifecXpujSvk8XU2ytJ
/JgMBxPIBKglN4eEMet9b4FRB5XeBdPAm19/LXyb4IlIipGNXlgNz/HCuBzidzHT
QQdqfA9rZVx1hwFr7AJCVqWaXVsx1oEAhKqpTtsLMyj594DvnRuwKw5Vse+1eydW
MIHYdbxmJccsTGIt/hsOpc8zfm+QYk5752jshhOKEBy+Ey3QZI1WbO547NOb2Hwr
Pgt7fw2NCKMPElSu98zmneFPhqNHf7L5urBe5gADj81E8lm6t/oVxcLBfAQYAQoA
JgIbDBYhBP65IJ8vLz9GZIQeVawQezhmktrdBQJnABcLBQkRa8qFAAoJEKwQezhm
ktrde3MP/13CLWp99XvRROrzD/bWOfWjAenT2PE/tYdOY9YcTQFbnIUhaVUDWAo3
pibR3D4u9LlY4olpGfJ7BTIHFa9myfpaVvmrNjueYI4omli24JQ/CKqNdY8Qzxz+
/QyiNK7Aw5cEBWIu84WGB1SsefWWT3rZe9YBb77gNcWHZ15pXTXrcgUxGY4808MC
I9YFWq8EA0iHawtFnmB3UFfClWt37Hy3PKvr1is3uG60+ULI8RQz3/+ZwSG8U+xt
b+I7H9+gITc1eFCb+tIwp5xWflyxcFXYk6UzOL7y3Fg2tIEuSNtIHUC9NDVobf6c
I0KAzZcMvKiPQiuBnVOjgDLmCZM5H6axj9x+gi4oVh6ea3HLqMzyjm5JkeCGgKWv
H0gD3yGEZDvcbavkQOle5T+4JefndKzCPrluX0iyx+oQiiOL8WieSSkSB6BsZcUN
SeuGJwM79Y7Oqld/YVrQNBZj5Vz+m3nZ+0EWDDMI0hRgMpSEIc+dnTC0u103Z+Rc
c2IJq8INmU653sUcfCZE12ParW4rF7ib6kViYrABT8f4e2TP0aOyP5kp51ied9qL
azaBA6tt/C9X1V2EJZK4srXtmcZO2Im45RAiVXyfpBAmmiF3eZWCbKe7qBC4rDRh
LZG4RQW/S86Da0BID7gQz9IFSkaG504MsDhvnA7iAqaHUHUepCsiwsF8BBgBCgAm
AhsMFiEE/rkgny8vP0ZkhB5VrBB7OGaS2t0FAmUkSiQFCQ+P/Z8ACgkQrBB7OGaS
2t3AwA/9GkXKUgvjKGCxwE4SdDt7c2jw6to2TTP9iFJ3Xbk3+5BURT3gkZCuu9D7
gt+97aVo/B4EM7Xz8DQKyY7Ic9VAwDRra/Hwi1V0hw1zyIWQ/gAnX3baU6qLRWHR
vVR5meV8r35C+rg9DaWFYmvS7PIv9LfxESwBPUjbmx8k4/5EJpHUwf12bzkTnot5
7q5lHxKQa6IvqQak+Hp9ZM2KPdsgKO2HWJJIIvYcI5byW9zBKV0O7YR8gtRAJKp9
IbtsXx0WT6cqHOFVc5SSzdcaMt0gLFl7BTnJyvKK2l9GABGBmzYDjeCyF2J+Ippf
oqxqfTe6EoOsuEMc2PbLTs9SsWjyCC2VGlX8+uUH9SoKwL0VQ6LFsP6fhkVKqi/a
rB6UuPR/iZnrKIuxMNQ4U+t2Q6UdMlmXsAXTNdkwzoK9oJRokIrH0ZV1KtH4sjjA
tCic+tOddq+GQLiKe2WpJfxlA0uESCB0TxjAwQmfn1H+dUhPeLlbNimHlH0/hXPd
ifuNGozzADIRseQDyzjl8xGL1qRZLD3cfmda6RyZ+S3dQRuaRrcFCDccpY/pO+F8
jbx64zyqqNs+KV+SkQGOcKFhWTZGCfQ/zMDtDmQKjb3eTAkv1zdEOMw9zEjjmS0q
8FNl+2wO3VnvXwvBbtDdVCIaIq+jVcsy5XtnnV+bJ19Q9yue/XvCwWUEGAEKAA8C
GwwFAmEyoZoFCQueVRUACgkQrBB7OGaS2t1uHBAAhOYVvrtchRmzCvdNER1DtkIs
bgQPJ9OxbyfvmvoD06qxH7PrycLZKbt7yYpAUU/CMc86GwaEe0I5Nm1CTs6NvDIv
g3e7EPIS859tyQflbM56NlwbsopCuoCJYknuroIf/M6dW6vJKNXLMmnL/AtalUBw
X+5pblmGUUJep49oTOxQEnvnuqyvaGjXgFXix5PVFJD2ed5NnQeFpvfCpc/ioNOj
z7ORO82j1ht5nWqPraXX5AYhQFM/kwR1cK4LV7gVDd/q+dfGYHzpxQ/HtyX/Lasi
N6I52QqA95SM1ZZLPFLaNh6EvnB7uC9pLCYS8nvilX7/cez5PFff1e1gXCOT0jv3
mJ2exLmXV0BbfKgjccFCxhrdRLtukfiDfJkySy1zdscnpfng8wJ3xKRv43cUTz7M
Z24OYNMqK26aJZVXEQUYjCwsBylY/F5wjYAwgwZ8yF5RFix28P/K8JsIHb3QrAJK
sNWQAb03ZWis3N3spR5M9Mw3VuDZ3WUXq7mxB5M3kpVoZ3vETU5cwTbADYNPf4Sw
BDK2uIVtxabezxSBtz0FcyYoF+OW8q7r4WvoyC9/+3GfnozZLJcEIVDk4W2pMW4A
UhG/6drKTm3HkSDWIDu7d1sHWMffLEYfUHtN5DKkDkGoPfHvZvu9teR5yLfUrPTf
ktihPn/JMrmwa9pwi8LCwWUEGAEKAA8CGwwFAmCavPcFCQsGcHIACgkQrBB7OGaS
2t0uaA//UWRaRiHEAKeRqBG/T2ak+XZJNu7QHfNgoUEAub9Zru8oPPXx2AJLcHEN
KWmeFlLxADdWOZs4Bm9oOew3VQnR/dBqjnXfob9Rc+eYUjA3rXazM/QrqcU8Syi3
MjNGUmjdL5aQF+IppAMgOBLG1TEnM7C5/PvrGJuYpGEnkKEwMK/GYhqg2V60pHEV
```

```
Pvs66mefJpCzbZSy56qtknSt6yBNWc14XgDX6VTn2kW4CV/3vVJUuvjvYs9SPyY8
mKEXa6QvUd3PcXv6RiWk4lGYuT1+jh2VkcFQ+JnUwv9TbKFB9b5jq1bvW9+LMDEl
YXux7pBP5RPk+OLpyiExIRFWhi3x7aMWOzQ+I9yuNTeYkTHiEAQRUhs/1Fh4oLgI
v9QZgC0mRSN3zm8plQdivs1ZlAosAqqkA9BQwqsgosQe7P92irYIJqay0si9wGCD
wSMsmeXdIF6wW3/UMJZl66aarPeiZApGX0QdTZwjMh/QK/8gTKyeZulKmNkNfwWq
O17OirWqLKssVHtg3VUM8EIdh+oNqDDXSeWtYUmpPpWp+yWZ0x1MFFZhUQHQZTGu
TIj4A92LQzbrfj/jXRvWm2SrJMivUoiDUn+qxKIpVwFlI5gVb+uyTFhw89PCkphr
JwRi052RLoU9yd6Ek46UH4XfZZWrZuzY+zzB7oqGONphLgi/h3DCwWUEGAEKAA8C
GwwFAl771b8FCQlniTUACgkQrBB7OGaS2t2/MxAAjoEGPdzavhsOlXdPCRd1D5QJ
r8T/NSEV2z1cp8ZvdrkjNF09TBP4qsBnKJiuvY1Iw7OGX9W2okvXxgJizE45v9MH
WEMz4hmIjmAfRwcqENgpOc1IY/T0/+kkCW8dB6d30J1kT0n2PCRzN9L5vPqZXGTG
mLvd9MOjH1256w4uxLb+e1HMDTCqEN1ppq9G+EAR/29q8JZWs1marbZZWxSWcg/E
1YYbNafzklgjq4CLh/j8AEWSvLr39zRy9uvQ/yqAKZ4K4aZfh/SPupGDvsD6ZK54
EPHxErQ7aiXTbUHtvwhxWLOP6WmxFA3Shr6L6YUb6jq+0PVliFC517g3mxFHJtwt
yXGNIKhmzmr0l9OlsHafulJ/9QPfK3Ce32SkPhW/11MYA8HzduMv5Arp7cBczXSP
EUTmNIVKv3gTjSQrzRhwhHmMuqyDZ/rXQQ1jl2sxIDjO4MUMvVjYKF+OCNm42gVs
8ca3/wN9ZNU6hyFWeKQDuCAqPPbT5GO/DKseFEwB+07wwyH1RXbyl0v4fneg605X
S7lqhNtw2p1hDL0HYHDiV+aPZ+LoOmX6+dmnqE6bQJaIlVb922KWmliO7F3DkqP7
0jFlhoE1gfiXWkxP4Gy8wOobNfEMgvz02djkGQy+oQqeNdIcZFZgzPTGKB/nVgpt
9CcRDWjPltFCd2e1FBbCwWUEGAEKAA8FAld1gAUCGwwFCQeGH4AACgkQrBB7OGaS
2t1PIQ//Qc5VYfBCxpaMysaPQ44wXPEZSjxIGZhhMGzb1UzzAEYOw+RgKN5nNTXq
L2KoOkOrGnKqZOKByMdXwIPH/rGwwEsbbIpopnibf5ic5B/+xCTIK+qLIwX2ZLuk
NhbL6Y+E+7DxMMh+KqBWHONKkgwVY+rFWOfoops839ABKvc9/Ry4/qqkcb40AzpD
l1iQJ5vK/DMuaDWxWeKXqJLIl3WMGPcPfheuBZL1u7LEEHYKMgzvpbF81WIn3MBo
8jvxf2/o+kMafSSDqgvOu6yu8GOhmScpCbRJn7jV/HrG+tM+zy48TN6/MkGWSR7q
TD34pqBjyatVfVl6dGD6xj/i/Emt5hZB6qXruCDH7AWMoNx+FkDubs4sc4PKysZU
Itya6KdQFo2UeYsNwZhdn6QwKhd85um4JUHJCY0mARvjsQgWXH/5MR40cow77bbE
vVq0XNd+QRVlyT42CEtnIUOFLeDVuZrum5Tuvvna6ImMDoi/z6QcNeL79XsY2m6I
QVRiHr1BDb/8JLkfnWiwL8GRv169Kf8unx0y5u1YBpcMYkyDD2+pnnk3TY0rR+8X
8goecaS8fbyu/Q48K85ZMD8wKW/bzLQ+tK9y8xed24u2QERftMhIw9b6f45Nrrf/
PhgV8RnuwUusSbdDe8kw3eYTmLdzD4kZc9K7SdO2CqT+hm//9JI=
=uGHC
-----END PGP PUBLIC KEY BLOCK-----
```

以前的密钥

> **Important**
>
> 在之前的密钥过期之前会创建新密钥。因此，在任何给定时刻，可能会有多个密钥有效。从构件创建之日起，密钥就用于对其进行签名，因此当密钥的有效性重叠时，请使用最近发布的密钥。

到期日期：2025 年 10 月 4 日

密钥 ID	0xAC107B386692DADD
类型	RSA
大小	4096/4096
创建时间	2016-06-30
到期日期	2025-10-04
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-----
Comment: Hostname:
Version: Hockeypuck 2.2

xsFNBFd1gAUBEACqbmmFbxdJgz1lD7wrlskQA1LLuSAC4p8ny9u/D2zLR8Ynk3Yz
mzJuQ+Kfjne2t+xTDex6MPJlMYpOviSWsX2psgvdmeyUpW9ap0lrThNYkc+W5fRc
buFehfbi9LSATZGJi8RG0sCCr5FsYVz0gEk85M2+PeM24cXhQIOZtQUjswX/pdk/
KduGtZASqNAYLKROmRODzUuaokLPo24pfm9bnr1RnRtwt5ktPAA5bM9ZZaGKriej
kT2lPffbBjp8F5AZvmGLtNm2Cmg4FKBvI04SQjy2jjrQ3wBzi5Lc9HTxDuHK/rtV
u6PewUe2WPlnxlXenhMZU1UK4YoSB9E9StQ2VxQiySLHSdxR7Ma4WgYdVLn9bOie
nj3QxLuQ1ZUKF79ES6JaM4tOz1gGcQeU1+UklgjFLuKwmzWRdEIFfxMyvH6qgKnd
U+DioH5mcUwhwffAAsuIJyAdMIEUYh7IfzJJXQf+fF+XfOCl6byOJFWrIGQkAzMu
CEvaCfwtHC2Lpzo33/WRFeMAuzzd0QJ4uz4xFFvaSOSZHMLHWI9YV/+Pea3X99Ms
0Nlek/LolAJh67MynHeVBOHKrq+fluorWepQivctzN6Y1NOkx5naTPGGaKWK7G2q
TbcY5SMnkIWfLFSougj0Fvmjczq8iZRwYxWA+i+LQvsR9WEXEiQffIWRoQARAQAB
zSxBV1MgU0RLcyBhbmQgVG9vbHMgPGF3cy1kci10b29sc0BhbWF6b24uY29tPsLB
lAQTAQoAPgIbAwULCQgHAwUVCgkICwUWAgMBAAIeAQIXgBYhBP65IJ8vLz9GZIQe
VawQezhmktrdBQJnABcIBQkRa8qDAAoJEKwQezhmktrdl1MQAIwEuDar3OTxkfTa
cPNKDnNzxaWqrxZ3FTQ+PyrHhQ6usxxrvDKJS+uCjE9bmWHVFU1R4yQNF+72lJdw
5UhX0u+ZgT9afApE65uAZuwLhPsz8upXT8C6VeKXh3shdw7qXi2hrwtM1a0Pls4O
Cs2C9rLUDMJTySrVDDVwpnaAB+8DcFrs9bIt5Q3gdOUatdzDvcb7QKh9jUvzCpbE
```

```
cInb1epDN5MRzowMR4iU2VV1RzLxCvm7CQSyXfgf0DFLkXWiknhOq9eINmytJFG/
ntFdiZFkNZ5hP709loywdfNrmqB6PsF8BPGFh8gKw1pjowrfHpv6cNIqShmA76LT
3OHViOlqGFB7obffq//eZGPR0oYJFDrOdD2CFRoHnP3N++AfkA4SRN7eXwyoz6Pk
Do9WNIEEkAcp6PGvv7AokogDo/4OqmxgC6fN+3BTOstWpv4FlD4NxOZWsTs49wxg
kPlCCVf8t75aZZkcjXng1eClZZQ5SBlRtSB7gMqtP7MIn2J5w8spNbs5xQvJc76u
NvzwEasPkY+UcHd05RddOUwoKqDerLUG7YqdlNCJoQR1mBIgZButbQlMyaZcmQqO
iR0kwDi9h6Dl6fnUb2dFCNJw+eDHvsjG8HI3IVZMl0bUQ2kmmwr1O2YQ1ynJQm0l
lMllI4hFU8/lHNHm8ie5darpVXQEwsGUBBMBCgA+AhsDBQsJCAcDBRUKCQgLBRYC
AwEAAh4BAheAFiEE/rkgny8vP0ZkhB5VrBB7OGaS2t0FAmUkSiIFCQ+P/Z0ACgkQ
rBB7OGaS2t3HVg//S+/KbeOBf+nCdHsrWtp9kxvWIpAGvQhIbxx1tp/impfm+5Rm
fKPD0KX+g42fuMmOdDE4gjO4GjGd7ZY3bx+0zbDSdVebzmYCbPZ/BDP990oPKidd
w6Gl8PaIyqfuARKOESBETvAwNgwO4t2ocjs4pYZV+CuHvESYpqkuHjmHtye6ajZW
Mv24NhjVo4EfP33dPugTjXLjeuGT7qQpsYV3a66juHmPVkXwuPqxh9wTNc5TU6FG
UPSfIGMPL0xha7Rg2i5zvRaAxx4bHqGO8IAz/l/E/tJkV5xnt494HQam9UDbiFI0
Q0TSve1R6S45/UjQW6cycyduHtk72s9ipa9YM0ilTdLgKMWFjzYv4h4qeYvLw3oB
JGQew+I0I4dIrwL/TKet33EuFfwmyT9MaJBhqV6geFaQ0uVmwvzpAcvxIoSqSpkJ
B/kASqCEM/o0QZLuWc56cDsmMisD0ouVPt+c1Zk7AWLlf6j8LKYTbK0QxLRh/eeZ
jhSf3HnpaCfonbOoHmeo5d/o3EZ0AiA4GbT3xgScoIgXOT7KGgOWmtWkdyd3Zvl/
o6q6Hwpg4RsqCkwnfNm5ZIvmTAYXWc2hiICSicxrP0fek5Cc4xVgJR5RMNGyI7+m
ut1SE2WvLhMCwEyl5ecWF0tUze8VBlWkHJp0Y4k2ado39Zq/DZrTRQYEVrjCwX0E
EwEKACcCGwMFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AFAmEyoZoFCQueVRUACgkQ
rBB7OGaS2t3axA//dgcz5T4Fs7LWdIQB/KRnvX64IzaGQcwt3FBhYH+sFaSaD+lu
752vi3jlGxMBKs2NFxk6e2U8xBEir3vfKYd+mZ5L6egXC9MYfvMO/UFEFH3A+t3V
dTOJK4RQcaL9+rFRVdDmZuifN9OFfy25d66JCZ5OiqgHTQViVmbrGw2cQdyNWXrq
YLgg+gktadkmzis4J6hF/le8NOBfrG3n+QthFl/v2ppYYW9pmmxzUIf6tAlR1Vr8
PhPukjuFfrPLRL3XPiK4LkdlSI5MX7Llq4RkcZN1NY1LWS8699wJOLRcr8aQYvzZ
Jm7tfZUaekJ5ScXJWJEaWT4poMbWxXINj6VwE+DqKWvjkzoxBXSIdLk4XThA1dIq
3n5SfMkfuWV2A2xHqJtEzI1XeTm/d/JRxIG8hjIs5FNMGJUSNANJuTVA2putCVf0
JbP4QlafXoEVOYwW/EFJY+brjQadwc/knf/QxszDcKb3THuTxR80A0h6ZysmtLEg
PCaD1xUCDdr4P7DGOtV7yMaDR6O8QKmb7TKzCKCPnHouqPT0DhSB2MRq437+mfSe
EGga/sPMxe0z8ugO2CnrCf3eplU3ZOy4eeQSThTKe0Hp5YlsF1cdZSvjt7GJaHR2
9A22nAJY9PojtlM+0Dkr/PH6r2brv3sEuACRNhzqCWhAve+zVnVLeb+Fk1rCwX0E
EwEKACcCGwMFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AFAmCavPYFCQsGcHEACgkQ
rBB7OGaS2t3fqg//asHYSTBI4IoLibSF5ZJ8NaLo4EqgRtsO0W1zr8fCoFbxI+yI
qWrNXR7eoFj8tWO7Tj6S0kQr4QZcucLeILfox6CtDzO3f3WQTH9m/0si5U4Jf3RA
gBd0vwxVSSSNEsIUUfylOBHLVwlhaTVfh1h1dZhzb18LA1Vqu0100GGxvaG3dUl4
oMSRSK8EeaSO4hIM8ScjVyhjsuPRm1jlwd1EXZ5mkHRGRMGmwi3XysJjIeQRFmuG
a9uPes7I/K85r7X91BLz+G/mkSZsrHxzLxF0mSZtQdhq08GyMeQlJavs7sb1UVbg
ag30JEAjgqRsNIQOMIvl2/amrRJ7DUyQu5YkZQsAyCIhSVZw6zlJlkYVKSw1Cu1I
VX1n/Aahx5wwaQZA1dDTolX8WvL9O/KmEGWbKUSov40uoRwSOeXP3QhW75kD4zT0
n3pSUc7tXka8GAz5Ar+r90l4wc9as2WjWADo3CX8cF0zQOrN1naMQ++xBIAG9ms8
y3L4ECoRqvjCpjUAfhflxwSM7uclCgFBINFKSLV+QGxzWfHjg3+bSQd0hrRn9j4z
Di9aJmFESQ6iykbUjiUFrcI1NUCFKz2awaxh+Sq8UkYcvuxljxdK/zYYEG8DSdQ2
uwlssHCjYVpAb16hq1EAaSqnhv86020G4Z6JCg3AUZt9RlMBpK2Pn/NmPSzCwX0E
```

```
EwEKACcCGwMFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AFAl771vAFCQlnimUACgkQ
rBB7OGaS2t35Jw/9GhpQquJVUCAsc4az7+ad8q2d9OUKXOLl2x0j3/ofS8umZJYr
8KXZxPqBqvLjlUyAB5Ur4fWVBdp9iP4Vj74jmRih7YatK8heGmNoUAnI1/90qV5O
ypFObfvWLxvcUNizmS/LYKdNuYcHwyw4bFyTz9gd3XH6Dxak7YiAXz3JgJIXcIB3
ELfpsduzpncLeVwz/dKhYX5iJ7Y/xd4Ew8Ian8FyNDNRwcXobTpPAMNiGLG5xSLq
8RgQfkm07BVVDtu5tPw4V46gnBXGXNjPhSirGMoNbKZqtPO7TcBQhPQ9c95x73hs
kAqZKHrwi1B7cfy1I8y5sRFbPa0RXeVWQKEMZdwLsFhCbEex6iXHP+rMK0nSJf0G
9lF2eJk140n8K7wzakOnjvN00VfN/9D+xD2lCXCzbYbaDXX2tYOd4GS72fW9M/zT
96tIH5UtMGMOICuUJjnL34EbzBuxwTEnJKhDGQH2P+eUzM9jmuCTRLLGw2P0Zuof
6GsSNLf+5pw3BIvEZw5PYT1N6i3ml9MQ7p9BWr7flCF8CU/xzyPN7TVb/677Be7V
SL1rNFLNi0IdqcbLJ63pTWaUGkCSqRnMfq3cIDlzNz2LoVvOO8VVmdtFRkhHN8hJ
h7CUXlaqBOfaJhV3FqA9G8sXmSN3r3pusZb13kfCKsJUZorThWxdaUBuUH3CwX0E
EwEKACcFAld1gAUCGwMFCQeGH4AFCwkIBwMFFQoJCAsFFgIDAQACHgECF4AACgkQ
rBB7OGaS2t1aKA//dBaXto7zUFRbJvLgcSE3hJ0ChYZj8Uuo7iYK26mVycyBFQJK
Iv2XYFIJMwK1Rx1Ja1jA6BIKE3aYyx2LVTYU1Hke826dhH+kV2qN9C6t/VmYpV4b
n/i64po7EzD17ykrm5gB+z47uCvyC79/r8Ouns1mTf/JUql/hYJj2hf1MDWr07/X
Wc1zBLjlT93tg/43Vgz1wFdrUOcNx1+bQGXKTOi4AXSp3UvCL+2YsQ2/JsPf7ZzZ
awSuUVkZJrllp87S1MhZeHmTrCHkV1FHJyudLJErcH0337Jpd8xDRek7Klrx2pAS
cKIiyeAMIk/G1OuExYqEEVFQzMr9ZlMNuhNBJAMr5hVGsTgrwy2hlIrBktXav07d
OleSlsqw9ViZ7F6t90x5l+NkwXVsLsGYSzuNyIfomNuoCgA+cfM3TjzVp41qsg1J
WJc9Bavaan2pKF6Lb9Fq8u3HZk2u+YZbvZkqkXwTdZZQOkEmoVV4Y1G86bdpmPyj
8eV7C02NxPii4l+qV8qJQu/6DsA0QwMtBMUNODm3BF2+ZmUHuhMGxq9/4vDE8heE
ffYhHtNftV6JwwzGZmeZkrYA1P9AGLeVp/6iNUe8H5/oPvh4s2rRmqN+L/dQU1ix
iOAT5iAKoRQGkduXrWc4fAY5KxDB9qna4oqXO6QP8rEflI8ELcNgYEj4oJvOwU0E
V3WABQEQALzM0Cs9Zvd08xOEvbEBj59LrS9d0HVKQ61gmkNakWC+jR35VD6FXpe6
UYAcBLrEbVYfKw9P0p6MhFKAsb570JoznKGzE1rVYUZQzhD0RKje35rvkajvEcjG
AWMLTjr87pWHeD0389ER64bzORncfa/l+YP56PI+CThb2wUvTTONGJkPQUpVhH+P
256cQL/Y0Fwu4XLerpwN+YKgMQ47raRcydobPeSfMQr9fVKRyOzFEOrvNpCVDUqi
77d0gLDLjHlIlDyOX5554S8XYLb91eYOiFvnu2pTCKiiExRCSYK29mAQePKlTCCn
QxOjbmBbGS8mVIkpQ5vpvXvzpY3JIjMXaDGqWSQSYGXhECyxCR5eOtKYbCwwPIc2
rIl5gW6yXyw9pKmj5XafTP7YHTvRSr7CZ/VLkDkWl6AfQ9nPOg1mjwjpDFpmN71h
JlSKMaZkh0QGV5FW3dK+GLwxiWdqx3htbZErWyvumWQF/xBF7puKJBEXcoM5KfkJ
uZekBwcnVkfNFF2RdkM1ALq8InGzLXc7ROuEm0BXVirfju7JRtWLb3UhJWCuhRW2
muyYegSTkag5MduD1IJK37GL8WIlAL65taYgZegUoxHdSaEOefOhspxuduz8d33z
UV1WCFhi+r/+BMCQmTRbF8ao7fTC1dGd084DRP6qE/dMT4u0ZEn7ABEBAAHCwXwE
GAEKACYCGwwWIQT+uSCfLy8/RmSEHlWsEHs4ZpLa3QUCZwAXCwUJEWvKhQAKCRCs
EHs4ZpLa3XtzD/9dwi1qffV70UTq8w/21jn1owHp09jxP7WHTmPWHE0BW5yFIWlV
A1gKN6Ym0dw+LvS5WOKJaRnyewUyBxWvZsn6Wlb5qzY7nmCOKJpYtuCUPwiqjXWP
EM8c/v0MojSuwMOXBAViLvOFhgdUrHn1lk962XvWAW++4DXFh2deaV0163IFMRmO
PNPDAiPWBVqvBANIh2sLRZ5gd1BXwpVrd+x8tzyr69YrN7hutPlCyPEUM9//mcEh
vFPsbW/iOx/foCE3NXhQm/rSMKecVn5csXBV2JOlMzi+8txYNrSBLkjbSB1AvTQ1
aG3+nCNCgM2XDLyoj0IrgZ1To4Ay5gmTOR+msY/cfoIuKFYenmtxy6jM8o5uSZHg
hoClrx9IA98hhGQ73G2r5EDpXuU/uCXn53Sswj65bl9IssfqEIoji/FonkkpEgeg
bGXFDUnrhicDO/WOzqpXf2Fa0DQWY+Vc/pt52ftBFgwzCNIUYDKUhCHPnZ0wtLtd
```

```
N2fkXHNiCavCDZlOud7FHHwmRNdj2q1uKxe4m+pFYmKwAU/H+Htkz9Gjsj+ZKedY
nnfai2s2gQOrbfwvV9VdhCWSuLK17ZnGTtiJuOUQIlV8n6QQJpohd3mVgmynu6gQ
uKw0YS2RuEUFv0vOg2tASA+4EM/SBUpGhudODLA4b5wO4gKmh1B1HqQrIsLBfAQY
AQoAJgIbDBYhBP65IJ8vLz9GZIQeVawQezhmktrdBQJlJEokBQkPj/2fAAoJEKwQ
ezhmktrdwMAP/RpFylIL4yhgscBOEnQ7e3No8OraNk0z/YhSd125N/uQVEU94JGQ
rrvQ+4Lfve2laPweBDO18/A0CsmOyHPVQMA0a2vx8ItVdIcNc8iFkP4AJ1922lOq
i0Vh0b1UeZnlfK9+Qvq4PQ2lhWJr0uzyL/S38REsAT1I25sfJOP+RCaR1MH9dm85
E56Lee6uZR8SkGuiL6kGpPh6fWTNij3bICjth1iSSCL2HCOW8lvcwSldDu2EfILU
QCSqfSG7bF8dFk+nKhzhVXOUks3XGjLdICxZewU5ycryitpfRgARgZs2A43gshdi
fiKaX6Ksan03uhKDrLhDHNj2y07PUrFo8ggtlRpV/PrlB/UqCsC9FUOixbD+n4ZF
Sqov2qwelLj0f4mZ6yiLsTDUOFPrdkOlHTJZl7AF0zXZMM6CvaCUaJCKx9GVdSrR
+LI4wLQonPrTnXavhkC4intlqSX8ZQNLhEggdE8YwMEJn59R/nVIT3i5WzYph5R9
P4Vz3Yn7jRqM8wAyEbHkA8s45fMRi9akWSw93H5nWukcmfkt3UEbmka3BQg3HKWP
6TvhfI28euM8qqjbPilfkpEBjnChYVk2Rgn0P8zA7Q5kCo293kwJL9c3RDjMPcxI
45ktKvBTZftsDt1Z718LwW7Q3VQiGiKvo1XLMuV7Z51fmydfUPcrnv17wsFlBBgB
CgAPAhsMBQJhMqGaBQkLnlUVAAoJEKwQezhmktrdbhwQAITmFb67XIUZswr3TREd
Q7ZCLG4EDyfTsW8n75r6A9OqsR+z68nC2Sm7e8mKQFFPwjHPOhsGhHtCOTZtQk7O
jbwyL4N3uxDyEvOfbckH5WzOejZcG7KKQrqAiWJJ7q6CH/zOnVurySjVyzJpy/wL
WpVAcF/uaW5ZhlFCXqePaEzsUBJ757qsr2ho14BV4seT1RSQ9nneTZ0Hhab3wqXP
4qDTo8+zkTvNo9YbeZ1qj62l1+QGIUBTP5MEdXCuC1e4FQ3f6vnXxmB86cUPx7cl
/y2rIjeiOdkKgPeUjNWWSzxS2jYehL5we7gvaSwmEvJ74pV+/3Hs+TxX39XtYFwj
k9I795idnsS5l1dAW3yoI3HBQsYa3US7bpH4g3yZMkstc3bHJ6X54PMCd8Skb+N3
FE8+zGduDmDTKitumiWVVxEFGIwsLAcpWPxecI2AMIMGfMheURYsdvD/yvCbCB29
0KwCSrDVkAG9N2VorNzd7KUeTPTMN1bg2d1lF6u5sQeTN5KVaGd7xE1OXME2wA2D
T3+EsAQytriFbcWm3s8Ugbc9BXMmKBfjlvKu6+Fr6Mgvf/txn56M2SyXBCFQ5OFt
qTFuAFIRv+nayk5tx5Eg1iA7u3dbB1jH3yxGH1B7TeQypA5BqD3x72b7vbXkeci3
1Kz035LYoT5/yTK5sGvacIvCwsFlBBgBCgAPAhsMBQJgmrz3BQkLBnByAAoJEKwQ
ezhmktrdLmgP/1FkWkYhxACnkagRv09mpPl2STbu0B3zYKFBALm/Wa7vKDz18dgC
S3BxDSlpnhZS8QA3VjmbOAZvaDnsN1UJ0f3Qao5136G/UXPnmFIwN612szP0K6nF
PEsotzIzRlJo3S+WkBfiKaQDIDgSxtUxJzOwufz76xibmKRhJ5ChMDCvxmIaoNle
tKRxFT77OupnnyaQs22UsueqrZJ0resgTVnNeF4A1+lU59pFuAlf971SVLr472LP
Uj8mPJihF2ukL1Hdz3F7+kYlpOJRmLk9fo4dlZHBUPiZ1ML/U2yhQfW+Y6tW71vf
izAxJWF7se6QT+UT5Pji6cohMSERVoYt8e2jFjs0PiPcrjU3mJEx4hAEEVIbP9RY
eKC4CL/UGYAtJkUjd85vKZUHYr7NWZQKLAKqpAPQUMKrIKLEHuz/doq2CCamstLI
vcBgg8EjLJnl3SBesFt/1DCWZeummqz3omQKRl9EHU2cIzIf0Cv/IEysnmbpSpjZ
DX8Fqjtezoq1qiyrLFR7YN1VDPBCHYfqDagw10nlrWFJqT6VqfslmdMdTBRWYVEB
0GUxrkyI+APdi0M2634/410b1ptkqyTIr1KIg1J/qsSiKVcBZSOYFW/rskxYcPPT
wpKYaycEYtOdkS6FPcnehJOOlB+F32WVq2bs2Ps8we6KhjjaYS4Iv4dwwsFlBBgB
CgAPAhsMBQJe+9W/BQkJZ4k1AAoJEKwQezhmktrdvzMQAI6BBj3c2r4bDpV3TwkX
dQ+UCa/E/zUhFds9XKfGb3a5IzRdPUwT+KrAZyiYrr2NSMOzhl/VtqJL18YCYsxO
Ob/TB1hDM+IZiI5gH0cHKhDYKTnNSGP09P/pJAlvHQend9CdZE9J9jwkczfS+bz6
mVxkxpi73fTDox9duesOLsS2/ntRzA0wqhDdaaavRvhAEf9vavCWVrNZmq22WVsU
lnIPxNWGGzWn85JYI6uAi4f4/ABFkry69/c0cvbr0P8qgCmeCuGmX4f0j7qRg77A
+mSueBDx8RK0O2ol021B7b8IcVizj+lpsRQN0oa+i+mFG+o6vtD1ZYhQude4N5sR
```

```
RybcLclxjSCoZs5q9JfTpbB2n7pSf/UD3ytwnt9kpD4Vv9dTGAPB83bjL+QK6e3A
XM10jxFE5jSFSr94E40kK80YcIR5jLqsg2f610ENY5drMSA4zuDFDL1Y2ChfjgjZ
uNoFbPHGt/8DfWTVOochVnikA7ggKjz20+RjvwyrHhRMAftO8MMh9UV28pdL+H53
oOtOV0u5aoTbcNqdYQy9B2Bw4lfmj2fi6Dpl+vnZp6hOm0CWiJVW/dtilppYjuxd
w5Kj+9IxZYaBNYH4l1pMT+BsvMDqGzXxDIL89NnY5BkMvqEKnjXSHGRWYMz0xigf
51YKbfQnEQ1oz5bRQndntRQWwsFlBBgBCgAPBQJXdYAFAhsMBQkHhh+AAAoJEKwQ
ezhmktrdTyEP/0HOVWHwQsaWjMrGj0OOMFzxGUo8SBmYYTBs29VM8wBGDsPkYCje
ZzU16i9iqDpDqxpyqmTigcjHV8CDx/6xsMBLG2yKaKZ4m3+YnOQf/sQkyCvqiyMF
9mS7pDYWy+mPhPuw8TDIfiqgVhzjSpIMFWPqxVjn6KKbPN/QASr3Pf0cuP6qpHG+
NAM6Q5dYkCebyvwzLmg1sVnil6iSyJd1jBj3D34XrgWS9buyxBB2CjIM76WxfNVi
J9zAaPI78X9v6PpDGn0kg6oLzrusrvBjoZknKQm0SZ+41fx6xvrTPs8uPEzevzJB
lkke6kw9+KagY8mrVX1ZenRg+sY/4vxJreYWQeql67ggx+wFjKDcfhZA7m7OLHOD
ysrGVCLcmuinUBaNlHmLDcGYXZ+kMCoXfObpuCVByQmNJgEb47EIFlx/+TEeNHKM
O+22xL1atFzXfkEVZck+NghLZyFDhS3g1bma7puU7r752uiJjA6Iv8+kHDXi+/V7
GNpuiEFUYh69QQ2//CS5H51osC/Bkb9evSn/Lp8dMubtWAaXDGJMgw9vqZ55N02N
K0fvF/IKHnGkvH28rv0OPCvOWTA/MClv28y0PrSvcvMXnduLtkBEX7TISMPW+n+O
Ta63/z4YFfEZ7sFLrEm3Q3vJMN3mE5i3cw+JGXPSu0nTtgqk/oZv//SS
=bboB
-----END PGP PUBLIC KEY BLOCK-----
```

到期日期：2024 年 10 月 8 日

密钥 ID	0xAC107B386692DADD
类型	RSA
大小	4096/4096
创建时间	2016-06-30
到期日期	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-----
```

```
xsFNBFd1gAUBEACqbmmFbxdJgz1lD7wrlskQA1LLuSAC4p8ny9u/D2zLR8Ynk3Yz
mzJuQ+Kfjne2t+xTDex6MPJlMYpOviSWsX2psgvdmeyUpW9ap0lrThNYkc+W5fRc
buFehfbi9LSATZGJi8RG0sCCr5FsYVz0gEk85M2+PeM24cXhQIOZtQUjswX/pdk/
KduGtZASqNAYLKROmRODzUuaokLPo24pfm9bnr1RnRtwt5ktPAA5bM9ZZaGKriej
kT2lPffbBjp8F5AZvmGLtNm2Cmg4FKBvI04SQjy2jjrQ3wBzi5Lc9HTxDuHK/rtV
u6PewUe2WPlnxlXenhMZU1UK4YoSB9E9StQ2VxQiySLHSdxR7Ma4WgYdVLn9bOie
nj3QxLuQ1ZUKF79ES6JaM4tOz1gGcQeU1+UklgjFLuKwmzWRdEIFfxMyvH6qgKnd
U+DioH5mcUwhwffAAsuIJyAdMIEUYh7IfzJJXQf+fF+XfOCl6byOJFWrIGQkAzMu
CEvaCfwtHC2Lpzo33/WRFeMAuzzd0QJ4uz4xFFvaSOSZHMLHWI9YV/+Pea3X99Ms
0Nlek/LolAJh67MynHeVBOHKrq+fluorWepQivctzN6Y1NOkx5naTPGGaKWK7G2q
TbcY5SMnkIWfLFSougj0Fvmjczq8iZRwYxWA+i+LQvsR9WEXEiQffIWRoQARAQAB
zsFNBFd1gAUBEAC8zNArPWb3dPMThL2xAY+fS60vXdB1SkOtYJpDWpFgvo0d+VQ+
hV6XulGAHAS6xG1WHysPT9KejIRSgLG+e9CaM5yhsxNa1WFGUM4Q9ESo3t+a75Go
7xHIxgFjC046/O6Vh3g9N/PREeuG8zkZ3H2v5fmD+ejyPgk4W9sFL00zjRiZD0FK
VYR/j9uenEC/2NBcLuFy3q6cDfmCoDEOO62kXMnaGz3knzEK/X1SkcjsxRDq7zaQ
lQ1Kou+3dICwy4x5SJQ8jl+eeeEvF2C2/dXmDohb57tqUwioohMUQkmCtvZgEHjy
pUwgp0MTo25gWxkvJlSJKUOb6b1786WNySIzF2gxqlkkEmBl4RAssQkeXjrSmGws
MDyHNqyJeYFusl8sPaSpo+V2n0z+2B070Uq+wmf1S5A5FpegH0PZzzoNZo8I6Qxa
Zje9YSZUijGmZIdEBleRVt3Svhi8MYlnasd4bW2RK1sr7plkBf8QRe6biiQRF3KD
OSn5CbmXpAcHJ1ZHzRRdkXZDNQC6vCJxsy13O0TrhJtAV1Yq347uyUbVi291ISVg
roUVtprsmHoEk5GoOTHbg9SCSt+xi/FiJQC+ubWmIGXoFKMR3UmhDnnzobKcbnbs
/Hd981FdVghYYvq//gTAkJk0WxfGqO30wtXRndPOA0T+qhP3TE+LtGRJ+wARAQAB
wsFlBBgBCgAPBQJXdYAFAhsMBQkHhh+AAAoJEKwQezhmktrdTyEP/0HOVWHwQsaW
jMrGj0OOMFzxGUo8SBmYYTBs29VM8wBGDsPkYCjeZzU16i9iqDpDqxpyqmTigcjH
V8CDx/6xsMBLG2yKaKZ4m3+YnOQf/sQkyCvqiyMF9mS7pDYWy+mPhPuw8TDIfiqg
VhzjSpIMFWPqxVjn6KKbPN/QASr3Pf0cuP6qpHG+NAM6Q5dYkCebyvwzLmg1sVni
l6iSyJd1jBj3D34XrgWS9buyxBB2CjIM76WxfNViJ9zAaPI78X9v6PpDGn0kg6oL
zrusrvBjoZknKQm0SZ+41fx6xvrTPs8uPEzevzJBlkke6kw9+KagY8mrVX1ZenRg
+sY/4vxJreYWQeql67ggx+wFjKDcfhZA7m7OLHODysrGVCLcmuinUBaNlHmLDcGY
XZ+kMCoXfObpuCVByQmNJgEb47EIFlx/+TEeNHKMO+22xL1atFzXfkEVZck+NghL
ZyFDhS3g1bma7puU7r752uiJjA6Iv8+kHDXi+/V7GNpuiEFUYh69QQ2//CS5H51o
sC/Bkb9evSn/Lp8dMubtWAaXDGJMgw9vqZ55N02NK0fvF/IKHnGkvH28rv0OPCvO
WTA/MClv28y0PrSvcvMXnduLtkBEX7TISMPW+n+OTa63/z4YFfEZ7sFLrEm3Q3vJ
MN3mE5i3cw+JGXPSu0nTtgqk/oZv//SS
=Z9u3
-----END PGP PUBLIC KEY BLOCK-----
```

文档历史记录

本主题介绍《AWS SDK for Java 开发人员指南》在其发展历程中的重要更改。

更改	描述	日期
the section called “当前密钥”	添加将于 2026 年 9 月 27 日到期的新 PGP 密钥。	2025 年 10 月 1 日
不支持的代码模式	记录迁移工具不支持的代码模式。	2025 年 9 月 23 日
迁移工具	GA 版本的更新。	2025 年 9 月 19 日
指标参考	添加有关 SDK 收集的指标的更多详细信息。	2025 年 9 月 3 日
the section called “使用 `VersionedRecordExtension` 实施乐观锁”	添加有关 DynamoDB 增强型客户端的 `VersionRecordExtension` 的详细信息。	2025 年 9 月 2 日
查找使用 1.x 客户端的应用程序	添加说明，以便在迁移到版本 2 之前通过查询 AWS CloudTrail 事件来帮助识别使用AWS SDK for Java 1.x 客户端的应用程序。	2025 年 8 月 20 日
Transfer Manager	为客户端构造函数、方法、模型对象和行为更改添加全面的迁移对照表。包括需要手动迁移的不受支持方法的详细代码示例。	2025 年 7 月 1 日
指标	添加全面的 `LoggingMetricPublisher` 文档。对指标主题的内容进行了重构，并改进了入门指引。	2025 年 6 月 20 日

更改	描述	日期
the section called “高级别更改”	在迁移指南中添加有关 DynamoDBMapper （v1）和 DynamoDB 增强型客户端（v2）之间空字符串处理差异的信息。	2025 年 6 月 18 日
目录重组	添加根据指南其他部分的内容整合汇编而成的“???”章节。	2025 年 6 月 16 日
???	添加有关如何使用 LegacyMd5Plugin ，以实现与那些仍要求 MD5 校验和的旧系统向后兼容的信息。	2025 年 5 月 19 日
the section called “S3 客户端差异”	添加AWS SDK for Java 的 v1 和 v2 之间的 S3 客户端差异信息，如果迁移工具无法自动迁移 V1 代码，则提供迁移示例。	2025 年 4 月 24 日
反序列化差异	添加适用于 Java 的 SDK v1 和 v2 之间的反序列化差异。	2025 年 4 月 10 日
the section called “SQS 自动请求批处理”	添加从适用于 Java 的 SDK v1 升级到 v2 时 SQS 自动请求批处理的迁移内容。	2025 年 4 月 8 日
???	更新有关自动校验和计算的信息	2025 年 4 月 3 日
the section called “配置 ALPN”	显示 ALPN 协议与基于 Netty 的 HTTP 客户端协商的配置。	2025 年 2 月 21 日
the section called “Lambda 函数”	添加有关使用 EMF 日志记录指标发布者与 AWS Lambda 捕获 SDK 指标的信息。	2025 年 2 月 6 日

更改	描述	日期
实现ContentStreamProvider	添加有关如何实施 ContentStreamProvider 的主题。	2025 年 1 月 29 日
the section called “使用校验和实现数据完整性保护”	更新内容：有关自动校验和计算的详细信息。	2025 年 1 月 16 日
the section called “使用 Amazon S3”	为使用 Amazon S3 的场景添加迁移内容。	2025 年 1 月 8 日
the section called “访问基于 AWS CRT 的 HTTP 客户端”	添加有关如何将特定于平台的 jar 与基于 AWS CRT 的组件一起使用的信息。	2024 年 11 月 14 日
the section called “使用 IAM Roles Anywhere 进行身份验证”	添加有关如何使用 IAM Roles Anywhere 进行身份验证的信息。	2024 年 11 月 6 日
the section called “缓存凭证配置示例”	添加一个使用 asyncCredentialUpdateEnabled 属性配置凭证提供程序的示例。	2024 年 11 月 4 日
the section called “使用自动请求批处理”	添加一个新主题，用于记录 Amazon SQS 的自动请求批处理 API。	2024 年 10 月 23 日
OpenPGP 密钥	更新当前 OpenPGP 密钥信息。	2024 年 10 月 10 日
the section called “在表达式中使用复杂类型”"and"the section called “更新包含复杂类型的项目”	添加有关如何在表达式和更新中处理复杂类型的内容。	2024 年 10 月 10 日
更新 Amazon S3 存储桶名称	更新 Amazon S3 存储桶名称。	2024 年 9 月 30 日

更改	描述	日期
the section called “使用基于账户的端点来优化性能”	为 DynamoDB 添加基于 AWS 账户的端点的相关信息。	2024 年 9 月 24 日
the section called “处理属性值为 bean、map、list 和 set 的对象”	更新与 DynamoDB 增强型客户端相关的部分，说明如何处理复杂类型的属性。	2024 年 9 月 6 日
the section called “配置服务客户端以进行快捷查找”	澄清在使用适用于 Java 的 Lambda SnapStart 时，`EnvironmentVariableCredentialsProvider` 的注意事项。	2024 年 8 月 19 日
the section called “配置并行传输支持”	添加了页面，其中包含有关如何启用并行传输支持的信息	2024 年 8 月 15 日
the section called “使用 AutoGeneratedUuidExtension 生成 UUID”	添加有关 DynamoDB 增强型客户端 `AutoGeneratedUuidExtension` 的信息	2024 年 8 月 14 日
???	添加有关迁移工具的部分（预览版）	2024 年 8 月 9 日
the section called “S3 事件通知”	添加相关部分，说明如何使用 S3 事件通知 API	2024 年 7 月 21 日
the section called “使用 DynamoDB”	添加 DynamoDB 映射/文档 API 的 v1 到 v2 迁移信息	2024 年 7 月 21 日
the section called “S3 事件通知”	添加 S3 事件通知 API 的 v1 到 v2 迁移信息	2024 年 7 月 21 日
the section called “重试”	添加重试策略主题	2024 年 6 月 18 日

更改	描述	日期
如何设置 JVM TTL	移除有关使用 java 命令行系统属性设置 networkaddress.cache.ttl 安全属性的说明。	2024 年 5 月 21 日
the section called “为 AWS Lambda 缩短 SDK 启动时间”	更新 HTTP 客户端建议以缩短 AWS Lambda 的启动时间	2024 年 5 月 14 日
the section called “指标参考”	重新组织指标表项目	2024 年 5 月 1 日
the section called “故障排除”	添加故障排除主题。	2024 年 4 月 26 日
the section called “每次请求收集的指标”	添加 SDK 报告的新指标。	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 更改为 5 秒。	2024 年 4 月 23 日
the section called “软件包名称到 artifactId 的映射”	在 Maven artifactId 映射主题中添加软件包名称。	2024 年 4 月 17 日
the section called “指标”	在指标部分添加配置详细信息。	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”	添加禁用 IMDSv1 的说明。	2024 年 3 月 14 日
the section called “分步说明”	添加分步迁移说明。	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 “配置基于 AWS CRT 的 HTTP 客户端”	添加有关基于 AWS CRT 的同步 HTTP 客户端的信息。	2024 年 1 月 5 日

更改	描述	日期
the section called “Amazon Cognito Identity”"and"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
OpenPGP 密钥	提供当前 OpenPGP 密钥。	2023 年 12 月 6 日
the section called “序列化差异”	描述 SDK for Java v1 和 v2 之间的序列化差异。	2023 年 12 月 5 日
the section called “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
???	添加了有关库和实用工具从适用于 Java 的 SDK v1.x 到 v2.x 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 `@DynamoDbIgnoreNulls` 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 `@DynamoDbPreserveEmptyObject` 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

更改	描述	日期
the section called “客户端建议”	从版本 0.23 开始，AWS CRT 支持基于 musl 的操作系统，例如 Alpine Linux。HTTP 客户端建议现在反映了 musl 支持。	2023 年 8 月 11 日
the section called “创建 IAM policy”	添加 IAM Policy Builder 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 客户端添加了 HTTP 代理支持信息和示例。	2023 年 6 月 2 日
重新整理了目录	将 代码示例 部分和调用 AWS 服务提升为顶级目录条目。	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 “分页”	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 增强型客户端 API	发布了重新编写的 DynamoDB 增强型客户端 API 主题。	2023 年 4 月 28 日
更新入门教程说明	修改了 Maven 原型，加入了 credentialsProvider 的选项；相应地修改了说明。	2023 年 4 月 11 日
the section called “客户端建议”	添加了如何确定使用哪种 HTTP 客户端的指导	2023 年 3 月 30 日

更改	描述	日期
IAM 最佳实践更新	更新了指南，使其符合 IAM 最佳实践。有关更多信息，请参阅 IAM 安全最佳实践。	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 “配置基于 AWS CRT 的 HTTP 客户端”	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”	添加了适用于 Amazon S3 实例元数据服务的 Java SDK 客户端指导示例。	2023 年 2 月 1 日
the section called “使用高性能 S3 客户端”	添加了有关基于 AWS CRT 的 S3 客户端的部分。	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 “最佳实践”	添加了最佳实践部分。	2022 年 11 月 18 日
the section called “使用外部进程”	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 “指标参考”	更新了 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 “为 AWS Lambda 缩短 SDK 启动时间”	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 “配置 HTTP 客户端”	添加了涵盖 SDK 中所有 HTTP 客户端的配置信息。	2022 年 10 月 26 日

更改	描述	日期
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 数据库服务和适用于 Java 的 SDK 2.x 的概述部分。	2022 年 9 月 13 日
EC2-Classic Networking 将停用	EC2-Classic 将于 2022 年 8 月 15 日停用。	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
the section called “传输层安全性协议（TLS）”	更新了 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 “Waiter”	发布了 Waiter；为新功能添加了主题。	2020 年 9 月 30 日
the section called “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 Amazon SNS 的示例主题。	2020 年 5 月 30 日
the section called “为 AWS Lambda 缩短 SDK 启动时间”	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 JVM TTL DNS 缓存主题。	2020 年 4 月 27 日

更改	描述	日期
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 “传输层安全性协议（TLS）”	将 TLS 1.2 添加到安全部分。	2020 年 3 月 19 日
the section called “订阅 Amazon Kinesis Data Streams”	添加了 Kinesis 流示例。	2018 年 8 月 2 日
the section called “分页”	添加了自动分页主题。	2018 年 4 月 5 日
???	添加了 IAM、Amazon EC2、CloudWatch 和 DynamoDB 的示例主题。	2017 年 12 月 29 日
the section called “Amazon S3”	添加了 Amazon S3 的 getobjects 示例。	2017 年 7 月 8 日
the section called “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) 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。